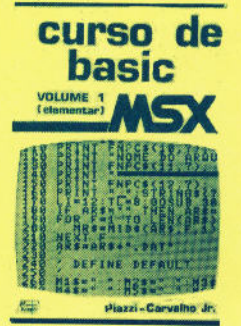
ANO 1 Nº 01
MAIO 1988
Cz$ 210,00

CPU

Gráficos impressora

Slots
Conheça o potencial do MSX

Sistema de Gravação

Copiador
Transfira seus arquivos de fita para disco e vice-versa

CPU

Águia Informática Ltda.
Rua Santa Clara, 98/415
Copacabana
Rio de Janeiro – RJ
CEP 22041
Tel. (021) 257-4402

DIRETOR RESPONSÁVEL
Gonçalo R. F. Murteira

DIRETORIA TÉCNICA
Antônio F. S. Shalders
Carlos E. A. Moreira
André L. F. de Freitas
J. L. Fonseca

REVISÃO DE TEXTO
Laura Maria Pinto

CAPA
José Aguilera

ASSINATURAS
Eduardo Simplício

ADMINISTRAÇÃO
José Newton Barros

Através da Revista CPU, uma publicação da Águia Informática, formulamos uma nova proposta na divulgação da informática.

Os assuntos serão tratados da maneira mais profunda e técnica, sem abrir mão de uma linguagem que seja acessível a todos.

Nosso objetivo é fazer com que todas as possibilidades do MSX sejam comentadas, a fim de que o usuário possa tirar o maior proveito deste equipamento.

Com o lançamento de CPU, lançamos no mercado, também, uma revista que irá divulgar projetos de hardware e software em conjunto. Serão projetos como cartão de 80 colunas, interface RS 232, etc.

Desde já, convidamos você a participar, enviando-nos suas descobertas, seus programas e dicas, bem como suas sugestões e críticas.

Com relação aos artigos publicados, informamos que os autores assumiram o compromisso de responder a todas as cartas que nos sejam enviadas. Portanto, em caso de dúvida, crítica ou sugestão, não deixe de nos escrever. Ela será bem recebida e você não ficará sem resposta.

GONÇALO MURTEIRA

ÍNDICE

SEÇÕES

Conexão MSX - impressora gráfica

ANTÔNIO F. S. SHALDERS

Um dos maiores desafios para o usuário da linha MSX é, sem dúvida alguma, a realização de uma cópia da SCREEN 2 em uma impressora gráfica.

Existem dois métodos de se fazer isso: o primeiro usa as tabelas da VRAM e o segundo testa diretamente os pontos na tela.

O objetivo deste artigo é fazer uma comparação em termos de velocidade e complexidade dos dois métodos para um determinado tipo de cópia.

Todas as rotinas apresentadas neste artigo foram desenvolvidas e testadas em um micro HOTBIT e em uma impressora Mônica da Elebra.

A LPTOUT

Esta é uma rotina importantíssima da ROM do seu MSX, pois é ela quem envia para a impressora o código do caracter a ser impresso. Para podermos utilizá-la corretamente, é necessário o uso de uma pequena subrotina em linguagem de máquina. E, mesmo que você não tenha experiência no assunto, irá entendê-la facilmente.

Existe um registrador do Z-80 chamado "acumulador" ou, simplesmente, registrador "A". A LPTOUT enviará para a impressora o código do caracter que estiver armazenado no acumulador, quando este for chamado através de um CALL, que é análogo a um GOTO do BASIC.

Para armazenarmos no acumulador um determinado valor, fazemos uso da instrução LOAD, que significa "carregar". A sintaxe desta instrução é LD A,XX. Feito isso, devemos chamar a rotina com um CALL relativo ao seu endereço de entrada, que, no caso da LPTOUT, é 00A5H.

Em último lugar, deverá haver o retorno ao BASIC, o que é feito quando a instrução RET (RETURN) for executada.

O nosso programa, em linguagem de máquina, ficará assim:

```
E000    3EXX      LD A,XX
E002    CDA500    CALL 00A5H
E005    C9        RET
```

A primeira coluna é relativa aos endereços; a segunda às instruções em hexadecimal; e a terceira, aos menemônicos do Z-80.

O endereço inicial da rotina foi definido com E000H por ser uma posição de memória relativamente alta e não requerer maiores cuidados na maior parte dos programas em BASIC.

Para chamarmos a rotina através do BASIC, basta definirmos em uma chamada USR seu endereço de entrada, por meio de um DEF USR.

Feito isso, basta pokearmos em E001H o código do caracter desejado e acessarmos a rotina através de uma declaração USR, como, por exemplo, um G=USR(0).

A CÓPIA

Para entendermos como o primeiro método funciona, é necessário fazermos um pequeno estudo da VRAM do MSX, quando em SCREEN 2.

Existem duas tabelas fundamentais para o nosso caso. Estas são a de padrões e a de cores.

A tabela de padrões contém a imagem propriamente dita, inicia no endereço zero e termina em 6143.

Os pixels do vídeo estão organizados de acordo com o esquema da figura 1, dispostos horizontalmente em octetos na seguinte ordem: o primeiro é formado pelos pixels relativos às coordenadas (0,0)...(1,7), até o oitavo, que representa a seqüência (7,0)...(7,7). O nono octeto é representado por (8,0)...(15,0) e assim sucessivamente até o último octeto, de número 6144, que corresponde ao grupo (248,191)...(255,191).

Cada um desses octetos pode ter somente uma cor, que está armazenada em uma tabela específica, que inicia em 8192 e, também, tem 6144 bytes de comprimento. Se mudarmos a cor de um único elemento de um octeto, a cor do octeto inteiro será modificada.

Aliás, este é um dos poucos defeitos do MSX!

O que esse primeiro método faz é justamente pesquisar essas duas tabelas e, após tratar corretamente os dados, haver a impressão.

Voltando a examinar a figura 1, podemos notar que os bytes relativos a cada octeto estão dispostos na horizontal, e isso nos gera um grave problema: quando a impressora entra em modo gráfico, ela é informada que lhe será enviada uma cadeia de n caracteres e estes deverão ser impressos no formato binário, que irá acionar uma determinada combinação de oito agulhas. O problema é que essas agulhas estão dispostas na vertical, sendo a agulha superior correspondente ao bit mais significativo e a inferior ao menos significativo.

A maneira mais fácil de contornar o problema é fazendo uma impressão na vertical. Basta fazermos uma varredura adequada da tabela de padrões e imprimi-

la. Examine o programa da listagem 1. A varredura usada é justamente para este tipo de impressão. A variável j contém o endereço de cada octeto a ser enviado para a impressora.

O programa da listagem 2 usa este algoritmo que agora lhes será explicado em detalhes.

As duas primeiras linhas do programa montam a nossa rotina em linguagem de máquina na memória, a partir do endereço E000H, definindo o endereço de entrada da chamada USR.

Na terceira linha, a cor de fundo da tela é armazenada na variável F% e a linha 60030 faz com que a impressora passe a trabalhar em modo EPSON, ajustando a entrelinha para oito pontos.

A linha 60050 faz com que a impressora entre em modo gráfico, esperando seqüências de 192 caracteres.

Os loops fornecem a seqüência correta de impressão dos octetos, sendo esses pokeados pelas variáveis J%, S1% e RP%, respectivamente.

É feita, então, a comparação com a cor de fundo, que decide ou não a impressão do octeto.

A chamada da LPTOUT é feita na linha 60090 e o CHR$(1) na linha 60110 faz o avanço da linha, assim que a seqüência gráfica tenha sido completada. A última linha faz com que a impressora volte ao modo normal de operação.

Observe que o programa da listagem 2 não copia SPRITES, pois, para isso, seria necessária a consulta das tabelas de formação e cores dos SPRITES.

Já pelo segundo método, as coisas ficam muito mais fáceis. Além de qualquer coisa ser copiada, a rotina é muito mais simples e objetiva. Basta fazermos uma varredura de todos os pontos do vídeo, através de dois loops e testar se a cor de fundo é igual à cor do ponto. Isso é feito com o auxílio de uma das funções menos utilizadas no BASIC – o POINT – e, nem por isso, é pouco eficiente.

Cada ponto aceso na tela irá disparar uma agulha e a entrelinha será alterada para 1 ponto. O teste dos pontos é feito na linha 60070 da listagem 3. O restante é idêntico ao programa anterior.

O primeiro método leva uma pequena vantagem quanto ao tempo de execução, mas convém citar que isto só é verdadeiro para uma impressão vertical e de dimensões 192x256.

A impressão horizontal já é bem mais difícil de se conseguir, pois o algoritmo de rotação da tabela de padrões é bem complexo, sendo a varredura radicalmente diferente para o primeiro método. Chamaremos cada grupo de oito octetos consecutivos de "grupo local" (Vide figura 2).

LISTAGEM 1

```
10 FOR C=0 TO 31
20 H=C*8
30 FOR L= 23 TO 8 STEP -1
40 K=256*L+H
50 FOR X=7 TO 0 STEP -1
60 J=X+K:PRINT J
70 NEXT X,L,C
```

LISTAGEM 2

```
50000 ' COPIA VERTICAL NORMAL (MSX)
50001 ' ANTONIO FERNANDO S. SHALDERS -
1988
60000 DEFINTA-Z:FOREN=0TO5:READCO:POKE&
HE000+EN,CO:NEXT
60010 DATA 62,0,205,165,0,201:DEFUSR=&H
E000
60020 F=PEEK(&HF3EA)
60030 POKE&HF417,1:LPRINTCHR$(27);"A";C
HR$(8)
60040 FORC=0TO31:H=C*8
60050 LPRINTCHR$(27);"K";CHR$(192);CHR$
(0);
60060 FORL=23TO0STEP-1:K=256*L+H
60070 FORX=7TO0STEP-1:J=X+K:SL=VPEEK(J)
:RP=VPEEK(J+8192)
60080 IFRPMOD16<>FTHENPOKE&HE001,255ELS
EPOKE&HE001,SL
60090 G=USR(0)
60100 NEXTX,L
60110 LPRINTCHR$(10);
60120 NEXTC
60130 LPRINTCHR$(27);"A";CHR$(13):POKE&
HF417,0
```

LISTAGEM 3

```
50000 ' COPIA VERTICAL NORMAL (MSX)
50001 ' ANTONIO FERNANDO S. SHALDERS -
1988
60000 DEFINTA-Z:FOREN=0TO5:READCO:POKE&
HE000+EN,CO:NEXT
60010 DATA 62,0,205,165,0,201:DEFUSR=&H
E000
60020 F=PEEK(&HF3EA):POKE&HF417,1
60030 LPRINTCHR$(27);"A";CHR$(1)
60040 FORXX=0TO255
60050 LPRINTCHR$(27);"K";CHR$(192);CHR$
(0);
60060 FORYY=191TO0STEP-1
60070 IFPOINT(XX,YY)=FTHEN60090
60080 POKE&HE001,1:GOTO60100
60090 POKE&HE001,0
60100 G=USR(0)
60110 NEXTYY:LPRINTCHR$(10);:NEXTXX
60120 LPRINTCHR$(27);"A";CHR$(13):POKE&
HF417,0
```

TABELA DE PADRÕES DA SCREEN 2

0 8 248
1
2
3
4
5
6
7 15 255

6136

6143

FIGURA 1

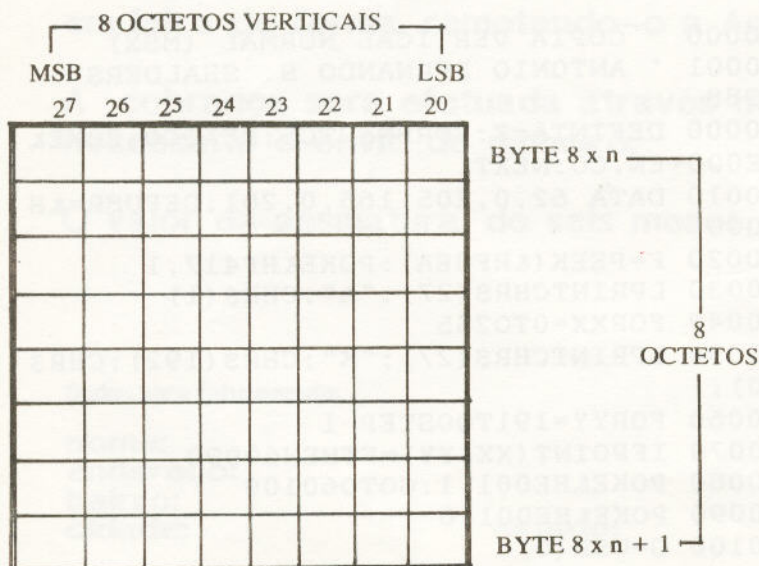



FIGURA 2

O nosso objetivo é tornar os octetos reais em octetos verticais, o que é conseguido agregando-se os bits de mesmo peso de cada um dos octetos reais: o primeiro octeto vertical será formado pela concatenação dos bits mais significativos dos octetos reais, e assim sucessivamente.

Se usarmos um método numérico para tal, a rotina se tornará extremamente lenta. Por isso, faço uso de um método alfanumérico, pois, além de ser muito mais rápido, é compacto.

O programa da listagem 4 faz exatamente isso. A matriz alfanumérica B$, de dimensões 8x1, contém as strings binárias relativas aos oito octetos, sendo transformada na variável U$ em um octeto vertical.

O resultado é armazenado no acumulador, sendo, posteriormente, impresso.

O programa da listagem 5 faz o mesmo tipo de cópia, usando o algoritmo do segundo método, sendo muito mais eficiente. Este programa não passa do mesmo programa da listagem 3 com os loops alterados.

A AMPLIAÇÃO

Para fazermos uma ampliação, o primeiro método se torna extremamente ineficiente.

No caso do primeiro método, se desejarmos uma impressão multiplicada por um fator de dois, teremos que ampliar um ponto dilatando-o na horizontal e na vertical.

A expansão na horizontal é trivial de ser feita, pois basta imprimirmos duas vezes seguidas cada octeto. Mas para ampliarmos na vertical, esbarramos em um problema: o código do octeto expandido passará a ter 16 bits e a impressora só tem oito agulhas. A solução é quebrarmos esse código em dois menores de oito bits, enviando primeiro para a impressora os oito bits mais significativos, e, em seguida, os oito menos.

O raciocíno é o mesmo para ampliações de fator 3 em diante (só que o primeiro método só permite ampliações múltiplas de 2). É como se a palavra "MSX" fosse transformada em "MMSSXX", "MMMSSSXXX", e assim por diante.

Os programas da listagem 6 e 7 fazem isso pelo primeiro método e os restantes pelo segundo.

O segundo método é muito mais adequado quando se trata de cópias que requerem algoritmos mais sofisticados e são muito mais rápidos e fáceis de serem compilados, pois até o compilador COMP32 é capaz de fazê-lo.

Por último, serão dadas tabelas para auxiliar a alteração do programa básico do segundo método para obtermos cópias ampliadas por um fator n e como fazer reversão de impressão para cada tipo.

LISTAGEM 4

```
50000 ' COPIA HORIZONTAL NORMAL (MSX)
50001 ' ANTONIO FERNANDO S. SHALDERS -
1988
60000 DEFINTA-Z:FOREN=0TO5:READCO:POKE&
HE000+EN,CO:NEXT
60010 DATA 62,0,205,165,0,201:DEFUSR=&H
E000
60020 F=PEEK(&HF3EA):POKE&HF417,1
60030 LPRINTCHR$(27);"A";CHR$(8)
60040 FORL=0TO23:K=256*L
60050 LPRINTCHR$(27);"K";CHR$(0);CHR$(1
);
60060 FORC=0TO248STEP8:H=K+C
60070 FORX=0TO7:J=X+H
60080 SL(X)=VPEEK(J):RP=VPEEK(J+8192)
60090 SL$(X)=RIGHT$(STRING$(8,"0")+BIN$
(SL(X)),8)
60100 IFRPMOD16()FTHENSL$(X)=BIN$(255)
60110 NEXTX
60120 FORX=0TO7:=X+1
60130 SC$=MID$(SL$(0),A,1)+MID$(SL$(1),
A,1)+MID$(SL$(2),A,1)+MID$(SL$(3),A,1)+
MID$(SL$(4),A,1)+MID$(SL$(5),A,1)+MID$(
SL$(6),A,1)+MID$(SL$(7),A,1)
60140 SC=VAL("&B"+SC$)
60150 POKE&HE001,SC:G=USR(0)
60160 NEXTX,C:LPRINTCHR$(10);:NEXTL
60170 LPRINTCHR$(27);"A";CHR$(13):POKE&
HF417,0
```

LISTAGEM 5

```
50000 ' COPIA HORIZONTAL NORMAL (MSX)
50010 ' ANTONIO FERNANDO S. SHALDERS -
1988
60000 DEFINTA-Z:FOREN=0TO5:READCO:POKE&
HE000+EN,CO:NEXT
60010 DATA 62,0,205,165,0,201:DEFUSR=&H
E000
60020 F=PEEK(&HF3EA):POKE&HF417,1
60030 LPRINTCHR$(27);"A";CHR$(1)
60040 FORYY=0TO191
60050 LPRINTCHR$(27);"K";CHR$(0);CHR$(1
);
60060 FORXX=0TO255
60070 IFPOINT(XX,YY)=FTHEN60090
60080 POKE&HE001,1:GOTO60100
60090 POKE&HE001,0
60100 G=USR(0)
60110 NEXTXX:LPRINTCHR$(10):NEXTYY
60120 LPRINTCHR$(27);"A";CHR$(13):POKE&
HF417,0
```

LISTAGEM 6

```
50000 ' COPIA HORIZONTAL AMPLIADA (MSX)
50010 ' ANTONIO FERNANDO S. SHALDERS -
1988
60000 DEFINTA-Z:FOREN=0TO5:READCO:POKE&
HE000+EN,CO:NEXT
60010 DATA 62,0,205,165,0,201:DEFUSR=&H
E000
60020 F=PEEK(&HF3EA):POKE&HF417,1
60030 LPRINTCHR$(27);"A";CHR$(8)
60040 A(1)=1:A(2)=5
60050 FORL=0TO23:K=256*L:FORT=1TO2
60060 LPRINTCHR$(27);"K";CHR$(0);CHR$(2
);
60070 FORC=0TO248STEP8:H=K+C
60080 FORX=0TO7:J=X+H
60090 SL(X)=VPEEK(J):RP=VPEEK(J+8192)
60100 SL$(X)=RIGHT$(STRING$(8,"0")+BIN$
(SL(X)),8)
60110 IFRPMOD16()FTHENSL$(X)=BIN$(255)
60120 NEXTX:FORX=0TO7:A=X+1
60130 SC$=MID$(SL$(0),A,1)+MID$(SL$(1),
A,1)+MID$(SL$(2),A,1)+MID$(SL$(3),A,1)+
MID$(SL$(4),A,1)+MID$(SL$(5),A,1)+MID$(
SL$(6),A,1)+MID$(SL$(7),A,1)
60140 X$=MID$(SC$,A(T),4):FORA=1TO4:C$=
MID$(X$,A,1):Y$=Y$+C$+C$:NEXTA
60150 Y$=RIGHT$(Y$,8):Y=VAL("&B"+Y$)
60160 POKE&HE001,Y:G=USR(0):G=USR(0)
60170 NEXTX,C:LPRINTCHR$(10);:NEXTT,L
60180 LPRINTCHR$(27);"A";CHR$(13):POKE&
HF417,0
```

LISTAGEM 7

```
50000 ' COPIA VERTICAL AMPLIADA (MSX)
50010 ' ANTONIO FERNANDO S. SHALDERS -
1988
60000 DEFINT A-Z:FOR EN=0 TO 5:READ CO:
POKE &HE000+EN,CO:NEXT
60010 DATA 62,0,205,165,0,201:DEFUSR=&H
E000
60020 F=PEEK(&HF3EA):POKE &HF417,1
60030 LPRINT CHR$(27);"A";CHR$(8)
60040 R(1)=1:R(2)=5
60050 FOR C=0 TO 31:FOR T=1 TO 2:H=C*8
60060 LPRINT CHR$(27);"K";CHR$(128);CHR
$(1);
60070 FOR L=23 TO 0 STEP-1:K=256*L+H
60080 FOR X=7 TO 0 STEP-1:J=X+K:SL(T)=V
PEEK(J):RP(T)=VPEEK(J+8192)
60090 SL$(T)=RIGHT$(STRING$(8,"0")+BIN$
(SL(T)),8)
60100 IF RP(T) MOD 16 () F THEN SL$(T)=
"11111111"
60110 X$(T)=MID$(SL$(T),R(T),4):FOR A=1
 TO 4:C$(T)=MID$(X$(T),A,1)
60120 Y$(T)=Y$(T)+C$(T)+C$(T):NEXT A
60130 Y$(T)=RIGHT$(Y$(T),8):Y(T)=VAL("&
B"+Y$(T))
60140 POKE &HE001,Y(T):G=USR(0):G=USR(0
)
60150 NEXT X,L:LPRINT CHR$(10);:NEXT T,
C
60160 LPRINT CHR$(27);"A";CHR$(13):POKE
 &HF417,0
```

LISTAGEM 9

```
50000 ' COPIA VERTICAL AMPLIADA (MSX)
50010 ' ANTONIO FERNANDO S. SHALDERS -
1988
60000 DEFINTA-Z:FOREN=0TO5:READCO:POKE&
HE000+EN,CO:NEXT
60010 DATA 62,0,205,165,0,201:DEFUSR=&H
E000
60020 F=PEEK(&HF3EA):POKE&HF417,1
60030 LPRINTCHR$(27);"A";CHR$(2)
60040 FORXX=0TO255
60050 LPRINTCHR$(27);"K";CHR$(128);CHR$
(1);
60060 FORYY=0TO0STEP-1
60070 IFPOINT(XX,YY)=FTHEN60090
60080 POKE&HE001,3:GOTO60100
60090 POKE&HE001,0
60100 G=USR(0):G=USR(0)
60110 NEXTYY:LPRINTCHR$(10);:NEXTXX
60120 LPRINTCHR$(27);"A";CHR$(13):POKE&
HF417,0
```

LISTAGEM 8

```
50000 ' COPIA HORIZONTAL AMPLIADA (MSX)
50010 ' ANTONIO FERNANDO S. SHALDERS -
1988
60000 DEFINTA-Z:FOREN=0TO5:READCO:POKE&
HE000+EN,CO:NEXT
60010 DATA 62,0,205,165,0,201:DEFUSR=&H
E000
60020 F=PEEK(&HF3EA):POKE&HF417,1
60030 LPRINTCHR$(27);"A";CHR$(2)
60040 FORYY=0TO191
60050 LPRINTCHR$(27);"K";CHR$(0);CHR$(2
);
60060 FORXX=0TO255
60070 IFPOINT(XX,YY)=FTHEN60090
60080 POKE&HE001,3:GOTO60100
60090 POKE&HE001,0
60100 G=USR(0):G=USR(0)
60110 NEXTXX:LPRINTCHR$(10);:NEXTYY
60120 LPRINTCHR$(27);"A";CHR$(13):POKE&
HF417,0
```

Tabela 1

ALTERAÇÕES DE TAMANHO
PARA A LISTAGEM 3

3X DE AUMENTO

```
60030 LPRINT CHR$(27);"A";CH
R$(3)
60060 LPRINT CHR$(27);"K";CH
R$(64);CHR$(2)
60080 POKE &HE001,7:GOTO 601
00
60100 FOR S=1 TO 3:G=USR(0):
NEXT S
```

4X DE AUMENTO

```
60030 LPRINT CHR$(27);"A";CH
R$(4)
60050 LPRINT CHR$(27);"K";CH
R$(0);CHR$(3);
60080 POKE &HE001,15:GOTO 60
100
60100 FOR S=1 TO 4:G=USR(0):
NEXT S
```

5X DE AUMENTO

```
60030 LPRINT CHR$(27);"A";CH
R$(5)
60050 LPRINT CHR$(27);"K";CH
R$(192);CHR$(3);
60080 POKE &HE001,31:GOTO 60
100
60100 FOR S=1 TO 5:G=USR(0):
NEXT S
```

6X DE AUMENTO

```
60030 LPRINT CHR$(27);"A";CH
R$(6)
60050 LPRINT CHR$(27);"K";CH
R$(128);CHR$(4);
60080 POKE &HE001,63:GOTO 60
100
60100 FOR S=1 TO 6:G=USR(0):
NEXT S
```

TABELA 2

ALTERAÇÕES DE TAMANHO
PARA A LISTAGEM 5

3X DE AUMENTO

```
60030 LPRINT CHR$(27);"A
";CHR$(3)
60050 LPRINT CHR$(27);"K
";CHR$(0);CHR$(3);
60080 POKE &HE001,7:GOTO
 60100
60100 FOR S=1 TO 3:G=USR
(0):NEXT S
```

4X DE AUMENTO

```
60030 LPRINT CHR$(27);"A
";CHR$(4)
60050 LPRINT CHR$(27);"K
";CR$(0);CHR$(4);
60080 POKE &HE001,15:GOT
O 60100
60100 FOR S=1 TO 4:G=USR
(0):NEXT S
```

5X DE AUMENTO

```
60030 LPRINT CHR$(27);"A
";CHR$(5)
60050 LPRINT CHR$(27);"K
";CHR$(0);CHR$(5);
60080 POKE &HE001,31:GOT
O 60100
60100 FOR S=1 TO 5:G=USR
(0):NEXT S
```

6X DE AUMENTO

```
60030 LPRINT CHR$(27);"A
";CHR$(6)
60050 LPRINT CHR$(27);"K
";CHR$(0);CHR$(6);
60080 POKE &HE001,63:GOT
O 60100
60100 FOR S=1 TO 6:G=USR
(0):NEXT S
```

TABELA 3

MÉTODO DE INVERSA
(IMPRESSÃO REVERSA)
LISTAGENS 3 E 5

TROCAR:

```
60080 POKE &HE001,1:
GOTO 60100
60090 POKE &HE001,0
```

POR:

```
60080 POKE &HE001,0:
GOTO 60100
60090 POKE &HE001,1
```

OBS: BASTA INVERTERMOS OS VALORES A SEREM ARMAZENADOS NO ACUMULADOR (POKES).

Os Vírus Binárius

ANDRÉ L. F. DE FREITAS

Esta seção se propõe a apresentar a vocês assuntos gerais, de uma maneira informal, sem a utilização de linguagem de programação ou outros artifícios que façam vocês, leitores, correr para o micro, apertar o botão de liga/desliga e perder o resto de suas vidas na frente do monitor de vídeo procurando uma solução para um problema ou digitando um programa.

Epa! Você, aí, que já ia ligar a CPU (apelido carinhoso que eu dei ao meu MSX)! Resista à tentação! Volte!

Bom, muito bom...

O que eu proponho aqui é discutir qualquer coisa interessante que, às vezes, não damos muita importância. Vejamos. Alguém já ouviu falar de um tal Pascal? Não a linguagem, mas um certo senhor Blaise Pascal. Qual foi o primeiro computador eletrônico construído? Quem descobriu o Brasil? Decerto não foi um microcomputador!

Vamos tentar falar sobre algo relacionado à informática, mas não com programas, rotinas, etc...

Não sei se vocês já ouviram falar do "VIRUS BINARIUS". Com o aparecimento do primeiro computador, este ser foi acidentalmente criado. Não seguiu os passos evolutivos de outras criaturas nem caiu do espaço num disco voador. Este pequeno ser, vejam vocês, não é nem mesmo constituído de matéria, é totalmente invisível, mas contaminou o mundo todo. Ele pode estar, agora mesmo, incubado no seu micro. Ou pior: você pode já estar contaminado por ele!!!

Quantos de vocês não perdem horas sentados à frente do micro digitando, jogando, ou mesmo olhando para o micro DESLIGADO, como se este fosse um troféu? Eu mesmo não pude resistir a escrever este artigo usando um processador de textos num MSX, vejam só!

Quanto é o fatorial de 5? Muitos vão responder:

– Espera um pouco...

Enquanto isso, as mãos se dirigem automaticamente para um teclado e, após alguns RETURNs, BACKSPACEs e outros mais, responderão:

– Cento e vinte.

Certo. Mas porque usar o micro?

A resposta é simples: não eram vocês, mas os VIRUS BINARIUS. Portanto, cuidado! Eles estão em toda parte, em todas as teclas, em todo o foton disparado de um monitor de vídeo para dentro dos seus olhos. Destino final: o cérebro humano. Aí não tem mais solução. A contaminação é total e pode causar muitos problemas com os amigos, com a "patroa", com os filhos e, até, com o papagaio que, de uma hora para outra, só saberá falar RUN, RUN, RUN...

A cura ainda não foi encontrada, mas dizem que, após uns dois anos longe do micro, um computador jogado fora, todas as listagens de programa incineradas e, quem sabe, uma lavagem cerebral, o VIRUS BINARIUS volta a incubar e hiberna por um bom tempo, só voltando à ação ao sentir a proximidade de um microcomputador. Às vezes, nem isso funciona, como é o caso de um amigo meu que nem mais sequer consegue andar em linha reta...

Isto tudo descrito acima é uma brincadeira. O VIRUS BINARIUS não existe (será?!), mas o que eu pretendo com isto é exatamente falar de coisas diferentes, de maneira agradável, como havia dito no início do artigo. Esta é uma revista com autores ainda não conhecidos pelo público, mas excelentes programadores e possuídores do nosso tão amado MSX. Pessoas como vocês, que se propuseram a criar uma coisa nova no nosso mercado. Uma revista dedicada a todos aqueles que amam a programação e, principalmente, o MSX. Espero que a idéia agrade e que esta seção possa ter grande utilidade na vida de todos vocês.

Daqui em diante, procuraremos explorar os mais variados assuntos e curiosidades.

Existindo ou não, o VÍRUS BINARIUS irá passear pelo seu computador, numa aventura emocionante, onde ele será o vilão e você o mocinho. O cenário não poderia deixar de ser outro: o seu computador e periféricos.

Aguarde os próximos números.

Transfira seus programas de disco para fita e vice-versa

MARCELO FONTOLAN

Grande parte dos usuários da linha MSX está passando seus arquivos de fita para disco, pois os discos oferecem maior rapidez e segurança no armazenamento de dados.

Quando o programa a ser gravado em disco for em Basic, não teremos dificuldades para efetuar a leitura da fita e gravá-lo no disco.

Em se tratando de programas escritos em assembler, a operação exige um pouco mais de cuidados e atenção, principalmente no que se refere aos endereços.

Visando facilitar este trabalho, publicamos um programa cuja função é ler um programa da fita cassete e gravá-lo no disco ou vice-versa.

Programas que estejam gravados em fita com proteção, ou seja, sem header, não poderão ser transferidos para o disco com o auxílio deste programas.

Pensando na necessidade que o usuário possa vir a ter, o programa também possibilita transferências de programas de disco para fita cassete.

O funcionamento do programa é bastante simples.

FUNCIONAMENTO

Ao completar o carregamento, é apresentado um menú com três opções, que permite selecionar qual a operação que deve ser efetuada.

CÓPIA DE FITA PARA DISCO

No menú, após selecionar a opção 1, prepare o gravador e pressione uma tecla. Será, então, lido o header do programa e serão apresentados na tela os endereços e os comandos para leitura do programa da fita e gravação no disco.

Conhecidos os endereços, retorne a fita até o início e posicione o cursor no comando para leitura, pressionando RETURN. Será iniciada a leitura.

Terminada a leitura, posicione o cursor no comando para gravação no disco, pressionando, novamente, RETURN.

O programa lido das fita encontra-se gravado no disco.

Alguns erros podem acontecer durante este processo, como erro de leitura (problema com a fita, volume, etc.), erro de gravação (disco protegido, não formatado, etc.) e caso o programa não seja do formato assembler.

Todo programa em BASIC tem sua área na memória a partir do endereço &h8000, sendo que alguns programas em assembler também possuem o mesmo início.

Neste caso, haverá a necessidade de carregar o programa após cada transferência.

A necessidadae de um novo carregamento será informado pelo próprio programa, através das mensagens "TECLE F4" e "TECLE F5", para os casos em que o arquivo copiado se sobreponha ou não, respectivamente, ao copiador.

CÓPIA DE DISCO PARA FITA

Ao solicitar transferência de disco para fita, será apresentado o diretório do disco e solicitado o nome do programa do qual se deseja efetuar a cópia.

Na tela, serão apresentados os comandos para leitura do programa do disco e gravação em fita, já com os endereços.

O cursor deverá ser posicionado no comando para leitura e depois no comando para gravação, prestando-se atenção para que o gravador esteja pronto para efetuar a gravação.

O PROGRAMA

O programa está dividido em subprogramas, a fim de facilitar a compreensão.

A rotina em linguagem de máquina efetua a leitura de header do disco ou da fita e está presente no programa através das linhas DATA.

Este programa, devido à sua função, só opera em sistemas que possuam unidade de disco.

```
0  ' Marcello Fontolan
1  ' Itajai, 25 de janeiro de 1988
2  ' HYPER COPY
3  ' Copiador disco-fita-Disco V1.0
10 ' Limpa tela
20 CLS:KEYOFF:SCREEN0:COLOR15,1,1
30 ' Define teclas de funcao
40 FOR F=1 TO 3: KEY F,"": NEXT F
50 FOR F=6 TO 10: KEY F,"": NEXT F
60 KEY 4,"RUN"+CHR$(34)+"HYPER"+CHR$(34
)+CHR$(13)
70 KEY 5,"RUN"+CHR$(13)
80 ' Apresentacao do Menu principal
90 PRINT " ----------------------------
---------- ": LOCATE 0,1: PRINT "!
                                  !": LO
CATE 0,2 : PRINT ".-------------------
------------------."
100 LOCATE1,1 : PRINT"------------- HYP
ER COPY---- V1.0 ----": PRINT : PRINT :
 PRINT
110 PRINT: PRINT TAB(10) "Digite:"
120 ' Opcoes do Menu
130 PRINT: PRINT TAB(5) "1 --> Copia de
 Disco p/ Fita"
140 PRINT: PRINT TAB(5) "2 --> Copia de
 Fita p/ Disco"
150 PRINT: PRINT TAB(5) "B --> Sair"
160 A$=INKEY$: IF A$="" THEN 160
170 ' Analiza opcao desejada
180 IF A$="B" OR A$="b" THEN END
190 IF A$="1" THEN 230
200 IF A$="2" THEN 880
210 GOTO 160
220 ' Copia de Disco p/ Fita
230 CLS
240 PRINT "-------- Copia de Disco p/ F
ita --------"
250 ' Apresentacao do diretorio
260 PRINT: FILES
270 ' Espera nome do arquivo
280 PRINT: PRINT : LINE INPUT "Digite o
 nome do arquivo: ";N$
290 ' Desvia programa caso tenha erro
300 ON ERROR GOTO 670
310 ' Abre arquivo
320 OPEN N$ FOR INPUT AS#1
330 ' Obtem dados sobre o arquivo
340 F=ASC(INPUT$(1,#1))
350 IF F<>254 THEN 790
360 X$=INPUT$(6,#1)
370 ' Fecha arquivo
380 CLOSE#1
390 ' Calcula parametros
400 CLS
410 PI=(ASC(MID$(X$,2,1)))*256+(ASC(MID
$(X$,1,1)))
420 PF=(ASC(MID$(X$,4,1)))*256+(ASC(MID
$(X$,3,1)))
430 PE=(ASC(MID$(X$,6,1)))*256+(ASC(MID
$(X$,5,1)))
440 TM=(PF-PI)+1
450 ' Transforma dados decimais em Hex.
460 TM$=RIGHT$("0000"+HEX$(TM),4)
470 PI$=HEX$(PI):PF$=HEX$(PF):PE$=HEX$(
PE)
480 ' Apresenta dados sobre o arquivo
490 PRINT "Arquivo :";N$
500 PRINT "Formato :";: IF F=254 THEN P
RINT"ASSEMBLER"
510 PRINT "Endereco Inicial : &H";PI$
520 PRINT "Endereco Final : &H";PF$
530 PRINT "Ponto de Execucao : &H";PE$
540 PRINT "Tamanho : &H";TM$
550 ' Apresenta comando de leitura
560 PRINT: PRINT: PRINT "Bload";CHR$(34
);N$;CHR$(34)
570 ' Apresenta comando de gravacao
580 PRINT: PRINT "Bsave";CHR$(34);"CAS:
";LEFT$(N$,6);CHR$(34);",&H";PI$;",&H";
PF$;",&H";PE$
590 PRINT:PRINT
600 ' Apresenta tecla a ser pressionada
610 IF PI<37000! THEN PRINT "Tecle 'F4'
." ELSE PRINT "Tecle 'F5'."
620 ' Coloca o cursor em posicao
630 LOCATE 0,7
640 ' Termina a execucao
650 END
660 ' Tratamento de erros
670 CLOSE#1
680 CLS
690 PRINT"--------- Tratamento de Erros
 ----------"
700 PRINT: PRINT
710 IF ERR=62 THEN PRINT " O Drive espe
cificado nao existe !"
720 IF ERR=56 THEN PRINT " O Nome do ar
quivo esta incorreto !"
730 IF ERR=60 THEN PRINT " O Disco esta
 com problema de formato !"
740 IF ERR=53 THEN PRINT " O Arquivo na
o existe !"
750 PRINT: PRINT " Tecle algo para cont
inuar ..."
760 A$=INKEY$: IF A$="" THEN 760
770 RESUME 20
780 ' Tratamento de arquivos nao BSAVE
790 CLOSE#1
800 CLS
810 PRINT"--------- Tratamento de Erros
 ----------"
820 PRINT: PRINT
830 PRINT " O Arquivo especificado nao
se encontra gravado com o BSAVE do MSX
DISK BASIC!"
840 PRINT: PRINT " Tecle algo para cont
inuar ..."
```

```
850 A$=INKEY$: IF A$="" THEN 850
860 GOTO 20
870 ' Copia de Fita p/ Disco
880 RESTORE 890
890 DATA CD,E1,00,D8,21,00,C0,06,10,E5,
C5,CD,E4,00,C1,E1,D8,77,23,10,F4,CD,E1,
00,D8,21,10,C0,06,06,E5,C5,CD,E4,00,C1,
E1,D8,77,23,10,F4,C9
900 ' Armazena dados em ASSEMBLER
910 FOR P=0 TO 42
920 READ A$
930 POKE (&HE050)+P,VAL("&H"+A$)
940 NEXT P
950 DEFUSR=&HE050
960 DEFFNPE(X)=PEEK(X)+256*PEEK(X+1)
970 ' Apresentacao
980 CLS
990 PRINT "-------- Copia de Fita p/ Di
sco --------"
1000 PRINT : PRINT "------ Prepare
o Gravador ------"
1010 PRINT : PRINT "--- E tecle algo q
uando pronto ... ----"
1020 A$=INKEY$: IF A$="" THEN 1020
1030 ' Leitura de parametros
1040 X=USR(0): MOTOR OFF
1050 ' Calculo do nome
1060 FOR R=(&HC00A) TO (&HC00F)
1070 P$=CHR$(PEEK(R))
1080 N$=N$+P$
1090 NEXT R
1100 ' Obtem formato
1110 F=PEEK(&HC000)
1120 IF F<>208 THEN 20
1130 ' Calcula parametros
1140 PI=FNPE(&HC010):'Ponto inicial
1150 PF=FNPE(&HC012):'Ponto final
1160 PE=FNPE(&HC014):'Ponto execucao
1170 TM=(PF-PI)+1
1180 ' Transforma dados decimais em HEX
1190 PI$=HEX$(PI):PF$=HEX$(PF):PE$=HEX$
(PE):TM$=RIGHT$("0000"+HEX$(TM),4)
1200 ' Apresenta dados sobre o arquivo
1210 CLS
1220 PRINT "Arquivo : ";N$
1230 PRINT "Formato : ASSEMBLER"
1240 PRINT "Endereco Inicial : &H";PI$
1250 PRINT "Endereco Final : &H";PF$
1260 PRINT "Ponto de Execucao : &H";PE$
1270 PRINT "Tamanho : &H";TM$
1280 ' Apresenta comando de leitura
1290 PRINT : PRINT : PRINT "Bload";CHR$
(34);"CAS:";N$;CHR$(34)
1300 ' Apresenta comando de gravacao
1310 PRINT : PRINT : PRINT "Bsave";CHR$
(34);N$;CHR$(34);",&H";PI$;",&H";PF$;",
&H";PE$
1320 PRINT:PRINT
1330 ' Apresenta tecla a ser pressionad
a
1340 IF PI<37000! THEN PRINT "Tecle 'F4
'." ELSE PRINT "Tecle 'F5'."
1350 ' Coloca cursor em posicao
1360 LOCATE 0,7
1370 ' Termina a execucao
1380 END
```

Gerando sons no MSX

ANTÔNIO F. S. SHALDERS

GERANDO SONS NO MSX

O seu MSX possui um processador de áudio modelo AY-3-8910, produzido pela General Instruments (EUA). Este integrado é do tipo LSI (Large Scale Integration) e, embora não seja o processador de áudio mais sofisticado do mercado, é um dos mais facilmente encontráveis, além de ser de fácil implementação e operação.

O BASIC possui uma instrução que permite trabalharmos diretamente com os seus 14 registradores, o SOUND, cuja sintaxe é: SOUND REGISTRADOR, VALOR.

Além do SOUND, existe a macro-linguagem PLAY, que nos permite fazer bons trabalhos, desde simples vinhetas até acordes.

O processador de áudio, ou simplesmente PSG (Programmable Sound Generator), possui 3 canais, com 8 oitavas cada, num total de 96 notas musicais disponíveis, além dos geradores de ruído e envoltória.

O PSG também controla a leitura de dados do gravador e as portas multi-uso de oito bits, que são as entradas para joystick.

COMO GERAR UM TOM:

Para gerarmos um tom de freqüência pré-determinada em um canal de áudio, devemos carregar nos registradores relativos aos ajustes de freqüência fino e grosso do canal em questão, selecionar a envoltória desejada, o volume de saída e a atuação ou não do misturador de canais.

O maior mistério para muitos é como calcular os valores necessários à carga dos registradores de ajuste de freqüência, o que é, na realidade, muito simples, pois basta aplicarmos a seguinte fórmula, a fim de obtermos o valor principal:

N = INT (C / F / 32),

onde C é o clock do seu computador, em hertz e F é a freqüência desejada.

Os clocks do Hotbit e do Expert são 3579545 Hz e 3575611 Hz, respectivamente, ou seja: aproximadamente 3.58 MHz.

Os valores que devem ser atribuídos aos registradores de ajuste fino (AF) e grosso (AG) podem ser facilmente obtidos pelas expressões abaixo:

AF = N MOD 256
AG = N 256

É bom notar que a resolução das freqüências obtidas é inversamente proporcional à freqüência de maneira exponencial, pois, à medida que aumentamos a freqüência, a resolução diminui.

Na faixa dos 100 Hz, por exemplo, a resolução chega a ser melhor que 1 Hz. Já na faixa dos 5 KHz, a resolução já cai para cerca de 400 Hz, logo não é muito lógico programarmos o PSG para gerar tons alternados de 5.000 Hz e 5.010 Hz, por exemplo, pois os dois tons gerados seriam idênticos.

A menor freqüência que pode ser gerada é de 4 Hz e a máxima vai além de 60 Hz, o que não adianta muito, pois a faixa audível vai somente até cerca de 16 KHz, se seu ouvido for muito bom!

A tabela 1 exibe os registradores do PSG e suas respectivas funções:

REG.	FUNÇÃO
0	A.F. canal A
1	A.G. canal A
2	A.F canal B
3	A.G. canal B
4	A.F. canal C
5	A.G. canal C
6	Ajuste de freq. centr. do ruído
7	Controle do misturador
8	Volume do canal A
9	Volume do canal A
10	Volume do canal C
11	A.F. da freq. da envolt.
12	A.G. da freq. da envolt.
13	Tipo de envoltória

Tabela 1: Os registradores do AY-3-8910.

MAIS SOBRE O PSG:

O PSG possui um gerador de ruído branco que pode ser usado em conjunto ou não com os três canais analógicos.

O chamado ruído branco é caracterizado por uma mistura aleatória de freqüências de amplitudes iguais ou não. É exatamente o chiado que aparece em um aparelho de televisão quando sintonizamos um canal livre. Efeitos muitíssimo interessantes podem ser conseguidos com o uso racional deste recurso do AY-3-8910.

Um ponto forte do nosso processador de áudio é, sem dúvida alguma, a capacidade de controle da envoltória. Mas o que é envoltória? A envoltória de um som (ou envelope, como dizem alguns) é a forma com que a intensidade ou amplitude varia em função do tempo

É graças às diferentes formas de envoltórias que podemos distinguir o som de uma guitarra do de uma flauta, mesmo que ambos sejam exatamente da mesma freqüência.

É claro que existem infinitos tipos de envoltórias, sendo estas responsáveis pelo timbre e nuances dos sons. A envoltória pode ser representada desde uma função constante até funções ultra complexas.

O nosso PSG não pode simular todos esses tipos de envoltórias, mas pode simular qualquer combinação de envoltórias da família "dente de serra", que inclui a triangular e a constante, gerando efeitos que chegam a ser estonteantes.

O som das ondas do mar, por exemplo, é caracterizado por um ruído branco, cuja envoltória é da família senoidal (na realidade, é uma sobreposição de várias senóides aleatórias), mas podemos obter ótimos resultados com a envoltória triangular!

Já o som de uma explosão ou de um sino tem como envoltória uma função exponencial inversa, do tipo Y = 1 / X, mas podemos obter resultados muito próximos disso com uma envoltória do tipo dente de serra, com a rampa negativa, o que caracteriza um início abrupto, seguido de um decaimento suave do som.

O registrador responsável por isso é o 13 e podemos escolher oito tipos de envoltórias de família dente de serra e constante, ou algumas combinações destes.

A tabela 2 mostra os tipos de envoltórias com as quais podemos programar o PSG.

As freqüências do gerador de ruído e do gerador de envoltórias também devem ser determinadas corretamente, a fim de obtermos resultados satisfatórios.

O método usado para tal é semelhante ao do ajuste dos canais analógicos de saída. Para o ajuste da freqüência dominante do ruído branco, não aconselho a usar fórmulas de espécie alguma, pois é melhor escolhermos "de ouvido" o que mais nos agradar. Os valores possíveis para carregarmos o registrador 6 variam de 0 a 31, sendo que, quanto maior este valor, mais grave é o tom dominante.

Já para o ajuste de freqüência da envoltória, necessitamos de uma fórmula do mesmo tipo da que mostramos para o ajuste dos canais analógicos de saída:

N = C / (1100 * F)

Com esta fórmula, podemos obter freqüências desde 0.1 Hz!

Os valores a serem armazenados nos registradores de ajuste fino e grosso são achados da mesma maneira descrita anteriormente para os canais analógicos.

É importante notar que o gerador de envoltórias está ativo somente quando o volume do canal está no máximo.

O CONTROLE MISTURADOR:

O objetivo deste controle é combinar, selecionar, ativar ou deativar os canais de som e os geradores de envoltória e ruído. Esta seleção é feita através do valor atribuído ao registrador 7 do PSG.

Cada bit deste valor tem uma função específica e, para a parte sonora, são usados apenas os bits de 0 a 5. Os bits 6 e 7 são usados na verificação dos estados das portas multi-uso.

Convém informar-lhes que os bits realtivos a este registro são ativos em zero, e suas funções são mostradas na tabela 3.

Se desejarmos habilitar os canais A e B com somm puro (sem ruído) e o canal C com ruído, devemos configurar o registrador 7 com o valor &B 00011100, ou seja: SOUND 7,56. Um canal pode ser usado para gerar tom e ruído simultaneamente, mas não podemos ter mais de duas envoltórias diferentes nos três canais. No caso de termos duas, uma terá que ser, obritoriamente, uma função constante, ou seja: o volume do canal em questão deverá ser menor qu e16.

Existem diversos editores musicais e sonoros para o MSX e alguns de altíssimo desempenho, como por exemplo, o SOUND (p/disco) e o SUPER SYNTH (na minha opinião, o melhor). Se você não possuir nenhum desses dois programas, não se desespere, pois o programa da listagem 1, se usado com bom senso, pode trazer resultados muito bons na elaboração de sons complexos.

Escrevam-nos enviando sugetões ou em caso de dúvidas, pois o objetivo desta revista é auxiliar o usuário da linha MSX.

REG. 13	ENVOLTÓRIA
0,1,2,3 e 9	
4,5,6,7 e 15	
8	
10	
11	
12	
13	
14	

Tabela 2: O registrador 13 e as envoltórias.

BIT	FUNÇÃO QUANDO EM ZERO
0	Habilita o canal A
1	Habilita o canal B
2	Habilita o canal C
3	Habilita ruído no canal A
4	Habilita ruído no canal B
5	Habilita ruído no canal C

Tabela 3: Funções dos bits 0 a 5 do registrador 7 do PSG.

```
10 REM
20 REM REVISTA CPU - MAIO 1988
30 REM ANTONIO FERNANDO SHALDERS
40 REM
50 REM GERANDO SONS NO MSX
100 KEYOFF:SCREEN0:COLOR15,1:CLEAR
110 PRINT "HOTBIT OU EXPERT (H/E)":PRIN
T
120 A$=INKEY$:IF A$="H" OR A$="h" THEN
C=3579545# ELSE IF A$="E" OR A$="e" THE
N C=3575611# ELSE 120
125 PRINT"* FREQUENCIA (4 a 16000)","*
VOLUME (0 a 16) (16 liga a envoltoria)"
:PRINT
130 PRINT "CANAL A:":INPUT "FREQUENCIA,
VOLUME ";FA,VA:PRINT
140 IF FA=0 THEN FA=4
150 PRINT "CANAL B:":INPUT "FREQUENCIA,
VOLUME ";FB,VB:PRINT
160 IF FB=0 THEN FB=4
170 PRINT "CANAL C:":INPUT "FREQUENCIA,
VOLUME ";FC,VC:PRINT
180 IF FC=0 THEN FC=4
190 NA=INT(C/FA/32):NB=INT(C/FB/32):NC=
INT(C/FC/32)
200 FA=NA MOD 256:GA=NA \ 256
210 FB=NB MOD 256:GB=NB \ 256
220 FC=NC MOD 256:GC=NC \ 256
230 INPUT "FREQ. DOMINANTE DO RUIDO (0-
31) ";FR:PRINT
240 INPUT "TIPO DE ENVOLTORIA ";TE:PRIN
T
250 INPUT "FREQ. DA ENVOLTORIA (MIN=.1)
 ";WE:PRINT:IF WE=0THENWE=.1
260 NE=C/(1100*WE):FE=NE MOD 256:GE=NE
\ 256
270 PRINT"TOM ATIVO EM A (S/N) ";
280 GOSUB690:IF A$="S" THEN TA=0 ELSE I
F A$="N" THEN TA=1
281 PRINT A$:PRINT
290 PRINT"TOM ATIVO EM B (S/N) ";
300 GOSUB690:IF A$="S" THEN TB=0 ELSE I
F A$="N" THEN TB=2
301 PRINT A$:PRINT
310 PRINT"TOM ATIVO EM C (S/N) ";
320 GOSUB690:IF A$="S" THEN TC=0 ELSE I
F A$="N" THEN TC=4
321 PRINT A$:PRINT
330 PRINT"RUIDO ATIVO EM A (S/N) ";
340 GOSUB690:IF A$="S" THEN RA=0 ELSE I
F A$="N" THEN RA=8
341 PRINT A$:PRINT
350 PRINT"RUIDO ATIVO EM B (S/N) ";
360 GOSUB690:IF A$="S" THEN RB=0 ELSE I
F A$="N" THEN RB=16
361 PRINT A$:PRINT
370 PRINT"RUIDO ATIVO EM C (S/N) ";
380 GOSUB690:IF A$="S" THEN RC=0 ELSE I
F A$="N" THEN RC=32
381 PRINT A$:PRINT
390 M=0:M=TA+TB+TC+RA+RB+RC:CLS
400 SOUND 0,FA:PRINT"SOUND  0,";FA
410 SOUND 1,GA:PRINT"SOUND  1,";GA
420 SOUND 2,FB:PRINT"SOUND  2,";FB
430 SOUND 3,GB:PRINT"SOUND  3,";GB
440 SOUND 4,FC:PRINT"SOUND  4,";FC
450 SOUND 5,GC:PRINT"SOUND  5,";GC
460 SOUND 6,FR:PRINT"SOUND  6,";FR
470 SOUND 7,M:PRINT"SOUND  7,";M
480 SOUND 8,VA:PRINT"SOUND  8,";VA
490 SOUND 9,VB:PRINT"SOUND  9,";VB
500 SOUND 10,VC:PRINT"SOUND 10,";VC
510 SOUND 11,FE:PRINT"SOUND 11,";FE
520 SOUND 12,GE:PRINT"SOUND 12,";GE
530 SOUND 13,TE:PRINT"SOUND 13,";TE
680 PRINTSTRING$(13,95):END
690 A$=INKEY$:IF A$<>"S"ANDA$<>"N"ANDA$
<>"s"ANDA$<>"n" THEN 690
691 IF A$="S" OR A$="N" THEN RETURN
700 IF A$="s" THEN A$="S":RETURN
710 IF A$="n" THEN A$="N":RETURN
```

EXEMPLO PRATICO:

Como já temos o editor de sons, falta somente entendermos como os sons são compostos, ou seja: A combinação adequada de freqüências, volumes, ruído e envoltória, afim de obtermos o som desejado, como por exemplo, o som do mar:

Este de som é caracterizado por ataque e decaimento suaves, modulando um ruído de média freqüência (ondas quebrando). A envoltória que nos proporciona este efeito é a de número 4.

Para ativarmos esta envoltória, é necessário que o volume do canal em questão esteja ajustado em 16. Feito isso devemos ajustar a freqüência do ruído, que no caso é cerca de 25. Devemos agora, ajustar os controles da envoltória e seu respectivo período (14 e .2), e selecionar o ruído ativo no canal "A".

```
CANAL A : 0,16
CANAL B e C : 0,0
FREQÜENCIA DO RUÍDO : 25
TIPO DE ENVOLTÓRIA : 14
FREQÜENCIA DA ENVOLTÓRIA : .2
TOM ATIVO NOS CANAIS : NÃO
RUÍDO ATIVO EM "A" : S
RUÍDO ATIVO EM "B" E "C" : N
```

No próximo artigo da série veremos como fazer um bom número de efeitos. Aguardem.

MÁXIMAS E MÍNIMAS

Programação estruturada

J. L. FONSECA

Esta é uma nova coluna, numa revista também nova e, como tal, vamos começar dizendo quais são os nossos objetivos.

Nesta coluna serão discutidas dicas e técnicas de programação de interesse do principiante e daqueles já mais avançados. Será uma coluna aberta a críticas e sugestões, que deverão ser enviadas à revista em nome desta coluna.

Hoje, começaremos discutindo sobre um assunto que, para muitos, parecerá óbvio, mas, ainda assim, é importante para o principiante e até para alguns que já não são tão principiantes.

Vamos, pois, falar sobre a programação estruturada. Esta técnica, tão em moda nos últimos anos, nada mais é do que um método de facilitar o trabalho do programador através da subdivisão de um programa em subprogramas mais simples, todos eles encadeados logicamente.

Muita gente pensa que só, se pode programar estruturadamente em Pascal, C, Forth, ou outra linguagem criada, desde a sua concepção, para este tipo de programação. No entanto, podemos programar estruturadamente em qualquer linguagem, até em assembler. Um bloco pode chamar, ou até conter, outros blocos, mas, uma vez chamado um bloco, este deve fazer a sua tarefa e retornar o comando ao que o chamou. Esta descrição é basicamente, a descrição de uma sub-rotina, só que, com a restrição de que o ponto de entrada da mesma deve ser único, e, embora possa ter diversas saídas, todas devem retornar ao mesmo ponto.

Os tipos básicos de blocos são os seguintes: decisão, repetição definida, repetição indefinida ou condicional e bloco linear. O primeiro é o equivalente em Basic a um bloco if... then... else, que executa uma de duas ou mais ações, dependendo de uma condição dada. O segundo é o equivalente à contrução for... next, que repete um conjunto de instruções por um número de vezes determinada. O terceiro seria um bloco que repete um grupo de instruções até que uma condição seja satisfeita. Finalmente, o bloco linear é qualquer grupo de intruções sem nenhum desvio. As descrições anteriores fazem menção a instruções, mas, o mesmo é válido, igualmente, para blocos completos, ou seja, um conjunto if... then... é considerado como uma única instrução, assim como uma instrução gosub o é.

O bloco deve sempre ser fácil de entender. Se um bloco está ficando muito complicado é hora de o subdividir em blocos mais simples. Do mesmo modo, o bloco deve ter apenas uma função e esta deve estar claramente definida. Estes blocos devem ser sempre precedidos de comentários explicativos e, nos pontos mais complexos, ter comentários específicos ao ponto em questão. Estes comentários podem parecer supérfluos, mas são a principal ajuda na hora de modificar ou depurar um bloco ou programa feito há algum tempo.

Uma outra vantagem de usar a programação estruturada é que os blocos são, em geral, independentes do programa como um todo e, assim, poderão ser usados em outros programas, facilitando o desenvolvimento. Podemos e devemos, pois, formar uma biblioteca com os blocos mais úteis para utilização posterior e, aqui, vemos as vantagens dos comentários e de programar usando blocos auto-suficientes.

Vamos, agora, dar exemplos do dito acima, com alguns trechos em Basic, sendo que tudo que for dito é válido para qualquer outra linguagem.

BLOCO DE DECISÃO

```
10 LET A=5: REM VARIAVEL QUE CONTEM O VALOR
20 REM O BLOCO SEGUINTE VERIFICA SE A QUANTIDADE EM A E SUPERIOR A UM VALOR DETERMINADO
40 IF A  >  LIMITE THEN PRINT "SUPERIOR" ELSE GOSUB 333
50 .
60 .
333 LET A=A+1
334 PRINT "VALOR DE = "; A
335 RETURN
```

Como podemos ver no trecho acima, a linha 40 executa uma de duas ações, dependendo de uma decisão, sendo que, após, qualquer delas continua no mesmo ponto, ou seja, na linha 50. Se houvesse necessidade de executar mais de uma instrução em qualquer das opções, estas deveriam ser agrupadas dentro de uma sub-rotina e chamada dentro do bloco de decisão, como podemos ver na instrução que segue o else. Verifique que todos os pontos importantes estão comentados no programa, o que nos permite entendê-lo a qualquer altura.

BLOCO DE REPETIÇÃO DEFINIDA

```
05 LET A=1
10 FOR I=1 TO 10; REM CALCULA OS FATORIAIS ATE 10
20 LET A=A*1
30 PRINT I,A : REM IMPRIME O NUMERO E SEU FATORIAL
40 NEXT I
```

No programa acima, o bloco entre as linhas 10 e 40 é repetido um número determinado de vezes. Esse número pode ser uma constante ou ser passado numa variável, mas é sempre conhecido na entrada.

No bloco de repetição indefinida, ao contrário do caso anterior, o número de vezes a repetir é desconhecido e depende de que se cumpra uma condição determinada. Essa condição pode ser testada no início ou no fim do bloco, dependendo do efeito desejado.

BLOCO DE REPETIÇÃO INDEFINIDA
TESTE DO INÍCIO

```
10 IF A=5 THEN GOTO 50 : REM TESTE INICIAL. SAI DO BLOCO SE SATISFEITA A CONDICAO
20 PRINT "APERTE UMA TECLA NUMERICA"
30 INPUT A
40 GOTO 10
50 PRINT "VOCÊ BATEU O NUMERO CERTO"
```

BLOCO DE REPETIÇÃO INDEFINIDA
TESTE NO FINAL

```
10 REM INICIO DO BLOCO
20 PRINT "APERTE UMA TECLA NUMÉRICA"
30 INPUT A
40 IF A < >5 THEN GOTO 10 : REM REPETE SE CONDIÇÃO NÃO VÁLIDA
50 PRINT "VOCÊ BATEU A TECLA CERTA"
```

Como se vê nos exemplos acima, cada bloco é uma unidade lógica independente e poderá ser usado em outros programas com poucas ou nenhumas modificações. Os exemplos dados podem servir de modelo para você criar os seus próprios blocos, modificando, apenas, as condições e as instruções abrangidas pelos mesmos.

Finalmente, para aqueles que estão achando que os comentários só ocupam espaço e que as sub-rotinas tornam o programa lento, aconselhamos a fazer o programa do modo indicado e, após estar pronto, remover os comentários de uma cópia que será usada para rodar.

As sub-rotinas devem ser colocadas no início do programa, pois, quando o Basic procura um número de linha, começa pela primeira linha do programa e continua a pesquisa, uma a uma, até encontrá-la e, assim, se as sub-rotinas estiverem no início, serão encontradas mais rapidamente.

Vamos, agora, experimentar o novo tipo de programação até a próxima edição, onde teremos novas dicas de como aproveitar melhor as linguagens disponíveis em nosso micro.

Até a próxima.

DICAS

CAPS LOCK

Muitos programas em Basic, ao solicitarem do operador uma entrada, só reconhecem o que foi digitado caso a tecla CAPS LOCK esteja pressionada, ou vice-versa.

Para contornarmos este problema, e evitar uma linha de programa maior do que o necessário, podemos fazer uso de uma variável do sistema, a CPAST, que indica o estado da tecla CAPS LOCK.

Caso CAPS LOCK esteja ativa, teremos em &HFCAB um valor maior que zero e menor que 255. Um valor igual a zero desativa a tecla.

Exemplo:

Poke &HFCAB,1 (ativa)

Poke &hFCAB,0 (desativa)

MÚSICA ALEATÓRIA

Como será a música no ano 3000?

O programa abaixo poderá dar-lhe uma dar-lhe uma idéia.

Verifique que há um certo padrão. A geração dos números aleatórios que estão sob a música é feita pelo micro segundo uma rígida regra matemática.

O programa Música Aleatória é parte integrante do livro 100 Dicas para MSX, da Editor Aleph.

```
100 PLAY "SOM8000","SOM8000","SOM8000"
110 L$="L"+STR$(INT(RND(-TIME)*31)*2+2)
120 X$=L$+"N"+STR$(INT(RND(-TIME)*60))
130 Y$=L$+"N"+STR$(INT(RND(-TIME)*30+50))
140 Z$=L$+"N"+STR$(INT(RND(-TIME)*16+80))
150 PLAY X$,Y$,Z$
160 GOTO 110
```

Slots e expansões

ANDRÉ L. F. DE FREITAS

Você já deve ter se indagado a respeito do sistema de SLOTS de seu MSX: como funcionam os cartuchos de jogos, interfaces, ou qualquer periférico conectado ao seu MSX via cartuchos. Neste artigo, procurarei esclarecer as dúvidas a respeito dos slots e páginas de memória que são um pequeno enigma na vida de usuários de micros padrão MSX.

O microprocessador Z80 é um processor de 8 bits capaz de endereçar 65536 de memória (64 Kbytes). De que forma, então, podemos ter 32 Kbytes de memória ROM e mais 64 Kbytes de memória RAM para uso no micro? Como podemos ter, no exterior, micros do padrão MSX com expansões de memória de 128 Kbytes? O responsável por isso é um circuito integrado chamado PPI, do inglês Peripheral Programable Interface, de identificação 8255. A descrição mais detalhada deste "chip", termo que usarei daqui em diante, ficará para uma outra ocasião, pois, no momento, só nos interessa o trabalho que ele realiza no micro.

Este chip é uma interface paralela contendo 4 portas de 8 bits cada. Uma destas portas, a qual chamaremos de porta A, é responsável pela lógica de seleção de slots no MSX. Cada grupo de dois bits desta porta pode conter um número entre 0 e 3, o qual vai indicar em que slot do MSX a página de memória correspondente vai estar ativa. Para entender melhor, observe a figura 1.

Como exemplo, se quisermos uma configuração semelhante a da figura 2, teremos o seguinte valor na porta A da PPI: &b00001010. Dividindo este número em quatro blocos de 2 bits, teremos: &b00, &b00, &b10, &b10. Em decimal, teremos 0, 0, 2 e 2, que, na ordem da primeira página até a quarta, significa que as duas primeiras estão ativas no slot 0 e as duas superiores no slot 2. Esta é uma das configurações de memória mais usada no MSX e o seu micro pode, inclusive, ter esta configuração.

Agora, vejamos, cada página de memória possui 16 Kbytes. Se temos 4 slots, cada um com capacidade de conter memória em 4 páginas, podemos ter 4x4=16 páginas de 16 Kbytes, perfazendo um total de 256 Kbytes de memória.

Cada página também pode ter uma expansão para mais 3 páginas iguais sobrepostas. Mas esta seleção não é mais tão simples como a primeira. Portanto, o seu micro poderia ter mais 3 blocos de memória de 256 Kbytes sobrepostos ao primeiro, o que daria um total de 1 Mbyte de memória. Ótimo, não é mesmo?

Aí, surge o problema de endereçamento de 64 Kbytes. Apesar de toda esta memória, linearmente, só podemos ter 64 Kbytes ativos.

Provavelmente, você deve estar odiando o Z80, mas não fique chateado. Você pode chavear páginas de memória mudando o valor da porta A da PPI, podendo acessar todas as outras páginas quando quiser. Mas lembre-se: nunca mais de 64 Kbytes simultaneamente.

Quando o seu MSX é ligado, os slots são pesquisados à procura de memória RAM. Este teste é feito das mais altas posições de memória para baixo. A medida que o sistema encontra RAM, vai habilitando esta memória para uso. Se houver mais de duas páginas contendo memória RAM entre os endereços &h8000 e &hFFFF, ele habilitará as páginas mais próximas ao slot 0. O seu MSX fica, então, com as páginas zero e um no slot zero contendo todas as suas rotinas internas de operação e o interpretador BASIC, que estão em ROM, nestas páginas, e a memória RAM livre nas páginas 2 e 3 em algum outro slot. A partir daí, ele executa uma inicialização em variáveis de sistema e outras funções prioritárias para, depois, entrar no interpretador BASIC.

Para saber qual a configuração de memória do seu MSX, entre com o pequeno programa em BASIC da listagem e rode-o. Ele mostrará quais as páginas ativas do seu micro, pois a configuração pode variar conforme o fabricante. O programa faz uma leitura na porta &hA8 do micro, a qual endereça a porta A da PPI. O valor correspondente lido é, então, dividido em grupos de 2 bits e passado para decimal, informando em que slot estão as páginas de memória do micro.

Agora que já sabemos como funciona a paginação de memória dos microcomputadores MSX, podemos conhecer um pouco mais do sistema de expansões.

Quem possui uma interface de disco, cartão de 80 colunas, ou mesmo um cartucho de jogos, já verificou que o mesmo é conectado a uma das entradas de cartucho do micro. Cada entrada destas contém um barramento e uma lógica de seleção correspondente a um dos slots do MSX. Quando um destes slots contém um periférico ou um cartucho de jogo, a inicialização do sistema procura um certo conjunto de bytes na memória correspondente a estas entradas. Estes bytes contêm informações úteis do sistema para que este reconheça o tipo de periférico conectado. Estas informações podem indicar se existe ROM no slot, o seu endereço de execução, se existe expansão de comandos a serem utilizados pela instrução CALL do basic, endereço de rotinas para manipulação de dispositivos, etc.

Como conclusão desta parte do artigo, vemos que o sistema de slots do MSX é algo de grande valor, pois, através dele, podemos manipular toda a memória e acessar um grande número de periféricos já existentes para a linha. Futuramente, em uma outra oportunidade, iremos explorar mais o acesso a periféricos. Trataremos, agora, somente da memória de

nosso microcomputador.

A seguir, veremos um programa que permitirá a vocês, usuários, explorar toda a memória "adormecida" do MSX.

Vamos supor que temos um programa em linguagem de máquina que não necessita do interpretador BASIC para ser executado. É o caso dos jogos em linguagem de máquina que você, provavelmente, possui. Vamos supor que o programa tem mais de 32 Kbytes de comprimento. Ora, se o micro só tem ativos 32 Kbytes de RAM, como poderemos carregar e executar este programa? Simples. O programa é, geralmente, dividido em blocos menores com 16 Kbytes cada. Ao se er o primeiro bloco com a instrução BLOAD, este bloco é carregado em uma página ativa do sistema e seu ponto de entrada é uma pequena rotina que verifica se há uma página de RAM correspondente à página do interpretador BASIC (página 1) em outro slot diferente do 0. Ao encontrá-lo, muda o valor da porta A da PPI para se configurar possuindo a página 1 em RAM. A seguir, o programa é realocado para esta página e a porta A da PPI recebe, novamente, a configuração original do sistema, retornando o controle do micro ao interpretador BASIC. Pronto. Temos 16 Kbytes de RAM contendo um programa em uma região da memória que não estava sendo utilizada; uma daquelas páginas que, a princípio, parecia não ter utilidade. Podemos fazer isto com vários blocos de 16 Kbytes de programa, desde que tenhamos páginas de memória suficientes para contê-las.

Outra limitação é a de nunca desativarmos a página 0 da ROM, pois lá estão contidas as rotinas básicas de operação do MSX. Se ativarmos a página 0 em outro slot, o programa lá contido deve realizar algumas destas funções, sob pena de perdermos o controle do micro. O mesmo é válido para a página 3, pois lá se encontram variáveis do sistema muito importantes para o perfeito funcionamento do micro. Ao final do carregamento dos blocos, o último deve conter uma rotina que manipule as páginas ocultas nos slots "vazios" do micro, e teremos, então, um belo jogo ou utilitário com mais de 32 Kbytes de comprimento. Para se rodar o programa, basta configurar a PPI para ativar a página correspondente ao bloco que queremos executar e pular para um endereço naquele bloco. Ao final de execução, podemos alterar a configuração novamente e executar outro bloco. Pode ser um pouco estranho ficar pulando de um lugar para outro, mas o resultado, quase sempre, é um ótimo jogo ou utilitário, o qual já fez com que muitos de nós perdêssemos algumas horas utilizando-o com grande prazer.

O programa da listagem 2 é uma pequena rotina em linguagem de máquina que procura uma página de RAM correspondente aos endereços &h4000 até &h7fff nos slots livres do seu MSX e realoca para lá um programa em assembler qualquer. Como exemplo de programa a ser realocado, temos o fornecido na listagem 3. Este programa somente imprime uma mensagem no vídeo sem a necessidade do interpretador BASIC. Na listagem 4 temos uma outra pequena rotina que só chamará o programa de impressão na página em que se encontrar e retornará ao interpretador BASIC.

Observando a listagem 2, veremos que a rotina começa tentando escrever um valor na memória e, depois, tenta ler de volta o mesmo valor. Este teste tem de ser feito duas vezes com valores diferentes, pois este endereço poderia ser em ROM e conter o valor testado, o que não nos levaria a nenhuma conclusão. Se for encontrada RAM, o programa passa a realocar o programa da listagem 3. Portanto, antes de rodar o programa da listagem 2, certifique-se de já ter digitado, também, o da listagem 3. Se o slot testado não puder ser usado, o programa continuará a procurar até encontrar uma área livre.

```
20 REM    SLOTS E EXPANSOES
30 REM    AUTOR: ANDRE LUIZ FREITAS
40 CLS
50 A = INP ( &HA8 )
60 A$ = BIN$ ( A )
70 FOR I=1 TO 8 STEP 2
80 B$ = MID$ (A$,I,2)
90 J = 3 - INT ( I/2 )
100 PRINT "PAGINA:"; J; "  SLOT: "; VAL
( "&H" + B$ )
110 NEXT I
```

```
110 REM SLOTS E EXPANSõES
120 REM AUTOR: ANDRE FREITAS
130 REM
140 REM ESTE PROGRAMA BASIC COLOCA NA
150 REM MEMORIA AS ROTINAS EM LINGUAGEM
160 REM DE MAQUINA CONTIDAS NAS LINHAS
170 REM DATA.
180 REM ESTAS ROTINAS CORRESPONDEM
190 REM AS LISTAGENS 2,3,4 DESTE ARTIGO
200 REM
210 REM
220 REM PRIMEIRA ROTINA - LISTAGEM 2
230 DATA D8,A8,32,F0,E0,21,00,40
240 DATA 06,03,DB,A8,C6,04,32,F1
250 DATA E0,D3,A8,3E,AA,77,56,2F
260 DATA 77,5E,7A,83,FE,FF,28,04
270 DATA 10,E8,18,08,21,00,E1,11
280 DATA 00,40,01,30,00,ED,B0,3A
290 DATA F0,E0,D3,A8,C9
300 REM SEGUNDA ROTINA - LISTAGEM 3
310 DATA 21,0F,40,7E,FE,00,28,06
320 DATA CD,A2,00,23,18,F5,C9,0C
330 DATA 53,4C,4F,54,53,20,45,20
340 DATA 45,58,50,41,4e,53,4f,45
350 DATA 53,07,00
360 REM TERCEIRA ROTINA - LISTAGEM 4
370 DATA 3A,F1,E0,D3,A8,CD,00,40
380 DATA 3A,F0,E0,D3,A8,C9
390 REM
400 FOR I=&HE000 TO &HE034
410 READ A$:A=VAL("&H"+A$):POKE I,A
420 NEXT I
430 FOR I=&HE100 TO &HE122
440 READ A$:A=VAL("&H"+A$):POKE I,A
450 NEXT I
460 FOR I=&HE200 TO &HE20D
470 READ A$:A=VAL("&H"+A$):POKE I,A
480 NEXT I
490 CLS:PRINT"ROTINAS CARREGADAS NA MEM
ORIA"
```

Você deverá utilizar um monitor assembler para entrar os programas na memória, mas, caso não possua um, não se desespere. Na listagem 5 eu forneço um programa em BASIC que colocará na memória, automaticamente, os programas 2, 3 e 4 que estão em instruções DATA nas linhas iniciais do programa.

Após os três programas estarem na memória, estamos prontos para rodá-los. Digite, no BASIC, as linhas abaixo:

```
DEF USR = &HE000
A = USR(0)
```

Pronto.

Agora, o programa de impressão já está em uma página de memória livre do seu MSX. Se você já digitou a listagem 4, estamos prontos para testar a paginação de memória do MSX. Repare que o programa da listagem 4 lê um byte do endereço &hE100 que contém a configuração da porta A da PPI para a página 1 ser em memória RAM. Esta configuração foi escrita ali pelo programa 2 para sabermos, exatamente, onde é esta RAM e não causarmos uma perda de controle do sistema. Digite, agora, no Basic, as seguintes linhas abaixo:

```
DEF USR = &HE200
A = USR1 (0)
```

Aí está a sua mensagem, sem que o interpretador BASIC estivesse ativo durante sua execução. O programa da listagem 4 alterou a configuração das páginas de memória, desativando o interpretador BASIC, chamou a rotina de impressão a qual usa uma chamada a uma rotina do BIOS para imprimir, escrevendo direto através do processador de vídeo e restaurou o sistema para que tivéssemos, novamente, o acesso ao BASIC.

Gostaram?

Espero que vocês não fiquem só por aí. As listagens são apenas pequenos exemplos do que se pode fazer com a paginação de memória do MSX. Deixo, aqui, a sugestão para que vocês alterem à vontade estes programas e se utilizem destas rotinas quando precisarem de "mais memória". Criem à vontade, pois, como vocês podem ver, o MSX é um micro novo no mundo e mais novo ainda aqui no Brasil, havendo, ainda, muita coisa a ser explorada.

Estarei à disposição de vocês para sugestões e opiniões, bastando entrar em contato com a diretoria técnica da revista.

Espero que todos tenham gostado deste primeiro artigo e, como já disse anteriormente, não ficaremos por aqui. Ainda há muito a se explorar. Aguardem.

```
; SLOTS E EXPANSÕES
; AUTOR: ANDRÉ FREITAS
; LISTAGEM 2

                    ORG 0E000H

DB A8               IN A,( 0A8H)     ; Lê configuração da porta A da PPI
32 F0 E0            LD ( 0E0F0H),A   ; Salva no endereço &HE0F0
21 00 40            LD HL,4000H      ; Endereço na página 1
06 03               LD B,3           ; No. de SLOTS a testar
DB A8      Loop:    IN A,( 0A8H)     ; Lê PPI
C6 04               ADD A,4          ; Incrementa bits correspondentes à
                                     ; página 1 na porta A da PPI
32 F1 E0            LD ( 0E0F1H),A   ; Salva no endereço &HE0F1
D3 A8               OUT ( 0A8H),A    ; Altera PPI
3E AA               LD A,0AAH        ; Valor a ser escrito na memória
                                     ; equivale a &B10101010
77                  LD ( HL),A       ; Escreve e
56                  LD D,( HL)       ; lê da memória
2F                  CPL              ; Complementa este valor. Se a primeira
                                     ; leitura for correta, o complemento
                                     ; será &B01010101
77                  LD ( HL),A       ; Escreve e
5E                  LD E,( HL)       ; lê da memória
7A                  LD A,D
83                  ADD A,E          ; Soma os dois valores lidos da memória
FE FF               CP 0FFH          ; Se há RAM no SLOT, o valor da soma
                                     ; é &HFF e o teste está correto,
28 04               JR Z,Achou       ; então salta para a rotina de
                                     ; transferência de blocos, senão
10 E8               DJNZ Loop        ; Decrementa B e volta ao loop de teste
18 08               JR Fim           ; Não achando RAM ao fim do loop, sai

21 00 E1   Achou:   LD HL,0E100H     ; Início do bloco a transferir
11 00 40            LD DE,4000H      ; Endereço de destino
01 30 00            LD BC,0030H      ; Tamanho do bloco
ED B0               LDIR             ; Transfere

3A F0 E0   Fim:     LD A,( 0E0F0H)   ; Carrega A com a configuração
                                     ; original da PPI
D3 A8               OUT ( 0A8H),A    ; Restaura valor na porta A da PPI
C9                  RET

; Ao final deste programa teremos:
;
; No Endereço &HE0F0 - A configuração original da porta A da PPI
; No Endereço &HE0F1 - A configuração para a qual a página 1
;                      está em RAM
```

```
SLOTS E EXPANSÕES
AUTOR: ANDRÉ FREITAS
LISTAGEM 3

                 ORG 0E100H
         CHPUT:  EQU 0A2H    ; Rotina de impressão do BIOS

21 0F 40         LD HL,400FH ; Carrega HL com endereço da
                             ; mensagem ( end. após realocar )
7E       Loop:   LD A,( HL)  ; Carrega A com valor de caracter
FE 00            CP 0        ; Compara com zero
28 06            JR Z,Fim    ; Se for zero encerra programa
CD A2 00         CALL CHPUT  ; Chama rotina de impressão do BIOS
23               INC HL      ; Avança caracter na mensagem
18 F5            JR Loop     ; Volta para o laço de impressão
C9       Fim:    RET

0C       Mensg:  DEFB 0CH
53 4C 4F 54      DEFM 'SLOTS E EXPANSOES'
53 20 45 20
45 58 50 41
4E 53 4F 45
53
07               DEFB 7
00               DEFB 0
```

```
; SLOTS E EXPANSÕES
; AUTOR: ANDRÉ FREITAS
; LISTAGEM 4

               ORG 0E200H

3A F1 E0   LD A,( 0E0F1H) ; Carrega acumulador com a
                          ; configuração de PPI salva
                          ; pelo programa da listagem 2
D3 A8      OUT ( 0A8H),A  ; Altera configuração da PPI
CD 00 40   CALL 4000H     ; Chama programa de impressão
                          ; na página 1
3A F0 E0   LD A, ( 0E0F0H); Carrega acumulador com a
                          ; configuração normal da PPI
                          ; salva peloprograma da
                          ; listagem 2
D3 A8      OUT ( )A8H),A  ; Altera configuração da PPI
C9         RET
```

SLOTS E PÁGINAS DA MEMÓRIA

VALORES DOS SLOTS CORRESPONDENTES
ÀS PÁGINAS DE MEMÓRIA

	SLOT 0	SLOT 1	SLOT 2	SLOT 3	PORTA "A" DA PPI	
PÁGINA 3	0	0	0	1	BIT 7	2 Bits correspondentes
	0	1	0	1	BIT 6	à página 3
PÁGINA 2	0	0	1	1	BIT 5	2 Bits correspondentes
	0	1	0	1	BIT 4	à página 2
PÁGINA 1	0	0	1	1	BIT 3	2 Bits correspondentes
	0	1	0	1	BIT 2	à página 1
PÁGINA 0	0	0	1	1	BIT 1	2 Bits correspondentes
	0	1	0	1	BIT 0	à página 0

FIGURA 1

EXEMPLO DE CONFIGURAÇÃO DE MEMÓRIA

	SLOT 0	SLOT 1	SLOT 2	SLOT 3	PORTA "A" DA PPI	
PÁGINA 3			▨		1 / 0	Pág. 3 no Slot 2
PÁGINA 2			▨		1 / 0	Pág. 2 no Slot 2
PÁGINA 1	▨				0 / 0	Página 1 no Slot 0
PÁGINA 0	▨				0 / 0	Pág. 0 no Slot 0

FIGURA 2

MATEMÁGICA

Autômato celular

J. L. FONSECA

Esta será uma coluna onde serão apresentados programas ou fragmentos de programas ligados à área da matemática recreativa, uma área por demais fascinante, apesar de pouco difundida entre nós.

Os nossos computadores podem não ser tão rápidos nem ter uma resolução tão alta quanto os usados em pesquisas nas universidades, mas são o suficiente para explorar alguns mundos e problemas bem interessantes. Ao longo dos meses, veremos artigos sobre gráficos de funções, criptografia, teoria dos números, lógica, autômatos, etc.

Hoje, apresentaremos um pequeno mas interessante programa sobre um organismo matemático conhecido como autômato celular. Este organismo vive num universo simulado no computador, tendo regras evolutivas próprias, definidas pelo programa. Esta classe de programas abrange um vasto campo de possibilidades, desde o programa aqui apresentado, com um universo unidimensional com células em 156 possíveis estados evolutivos, até universos multidimensionais, onde vivem células com centenas de estados.

Um dos mais famosos deste grupo de programas é o LIFE, com um universo bidimensional e células de dois estados, o qual será apresentado em outra ocasião.

O programa da listagem 1 cria um universo unidimensional, representado pelo vetor A%( ), o qual simula um círculo, ou seja, a sua última célula é adjacente à primeira. Cada uma das variáveis do vetor representa uma célula do universo e o seu valor representa o estado evolutivo da célula.

O programa representa as gerações sucessivas em linhas consecutivas da tela no modo 3, com as cores dos pontos representando os estados evolutivos e, deste modo, podemos ver 48 gerações de 64 células simultaneamente.

Para usar o programa, respondas às perguntas como indicado. O padrão inicial deve ser fornecido como pedido, sendo que as letras de A até P correspondem aos números de 0 a 15. Em seguida, responda, com um conjunto de 0s e 1s, às perguntas sobre as influências. Esta última pergunta é que governa como as células vizinhas influenciam a célual sendo testada no momento (0 – sem influência, 1 influencia). Neste programa, são consideradas células vizinhas às duas células imediatamente adjacentes à célula sob teste e às duas imediatamente seguintes a estas últimas.

O programa é simples e fácil de ser entendido e modificado para condições diferentes das dadas por mim. Sinta-se, pois, à vontade para modificá-lo e divertir-se com os belos padrões gerados por ele.

Escreva-nos dando as suas sugestões e críticas, pois as mesmas serão bem vindas. E até à próxima edição, onde veremos novas curiosidades.

```
140 DIM B%(3): ' INFLUENCIAS
150 DIM A%(1,63): ' UNIVERSO
160 A$ = "": B$ = "": X% = 0
170 KEY OFF: CLS
180 IF X% = 0 THEN GOTO 210
190 PRINT A$: INPUT "Deseja mudar o padrao ";B$
200 IF B$="N" OR B$ = "n" THEN GOTO 250
210 X% = 1
220 CLS: PRINT SPC(7);"Entre com o padr
ao inicial": PRINT "   ( ate 64 caracte
res entre A e P )": PRINT
230 LINE INPUT A$
240 IF A$ = "" THEN GOTO 220
250 FOR I% = 0 TO 63
260 A%(0,I%) = 0
270 NEXT I%
280 IF LEN (A$)>64 THEN A$=MID$(A$,1,63 )
290 FOR I% = 1 TO LEN(A$) - 1
300 A%(0,I%)=ASC(RIGHT$(A$,I%+1)) - 65
310 NEXT I%
320 PRINT: PRINT: PRINT "Escolha a infl
uencia das celulas vizi-  nhas (0/1)":
PRINT
330 INPUT "I - 2 ";B%(0)
340 INPUT "I - 1 ";B%(1)
350 INPUT "I + 1 ";B%(2)
360 INPUT "I + 2 ";B%(3)
380 ' VERIFICA E CORRIGE AS INFLUENCIAS
400 FOR I%=0 TO 3
410 IF B%(I%) <> 0 THEN B%(I%)=1
420 NEXT I%
430 SCREEN 3
450 ' INICIA A EVOLUCAO DAS CELULAS
470 FOR J% = 0 TO 47
480 E% = J% MOD 2: ' INDICE DA GERACAO ANTERIOR
490 F% = (J%+1) MOD 2: ' INDICE DA PROXIMA GERACAO
500 FOR I% = 0 TO 63
510 A% = (I%+1) MOD 63
520 B% = (I%+2) MOD 63
530 C%=(I%-1)MOD63:IFC%<0THENC%=63+ C%
540 D%=(I%-2)MOD63:IFD%<0THEND%=63+D%
550 A%(F%,I%) = (A%(E%,A%) * B%(2) + A%
(E%,B%) * B%(3) + A%(E%,C%) * B%(1) + A
%(E%,D%) * B%(0)) MOD 15
560 PSET (I%*4,J%*4),A%(F%,I%)
570 NEXT I%
580 NEXT J%
590 BEEP
600 B$=INKEY$: IF B$ = "" THEN GOTO 600
610 GOTO 170
```

O sistema de gravação cassete no MSX

ANDRÉ L. F. DE FREITAS

Todos os usuários de micros padrão MSX já estão familiarizados com o seu sistema de gravação. Estes micros se utilizam de um processo de gravação chamado FSK, do inglês Frequency Shift Keying, ou seja, chaveamento de freqüência. Aquele som que ouvimos quando acionamos o gravador com uma fita cassete contendo um programa e uma seqüência de pulsos em freqüências diferentes correspondentes a bits zeros e uns. Estas freqüências são entre 1.200 hertz e 4.800 hertz, dependendo da velocidade de gravação. Podem ser utilizadas outras freqüências e, de certa forma, isto é fácil de se conseguir, alterando algumas variáveis do sistema MSX. Mas não é aconselhável, pois os circuitos de áudio do MSX possuem filtros que eliminam freqüências superiores, o que reduz a confiabilidade da gravação.

O que a maioria dos usuários não sabe é o que está por trás de uma simples instrução SAVE e como isto é processado pelo micro a nível de máquina. Pretendo, neste artigo, dar uma pequena visão do que se esconde por trás de uma instrução BASIC de operação cassete.

Ao se salvar ou carregar um programa em cassete, o seu micro se utiliza de várias rotinas contidas nos primeiros 16 Kbytes de ROM, que vamos chamar de BIOS. Ali se encontram todas as rotinas básicas de operação do MSX, desde o controle do teclado, vídeo, som, até mesmo a operação em cassete.

```
  0 REM LISTAGEM 1
110 REM   O SISTEMA DE GRAVACAO CASSETE
 NO MSX
115 REM REVISTA CPU - MAIO 1988
120 REM   ANDRE LUIZ FRANCO DE FREITAS
130 DATA CD,EA,00,21,00,E2,7E,FE
140 DATA 00,28,08,E5,CD,ED,00,E1
150 DATA 23,18,F3,CD,F0,00,C9
160 REM CARREGA O PROGRAMA ASSEMBLER NA
 MEMORIA
170 FOR I=&HE000 TO &HE016
180 READ A$: A=VAL("&H"+A$)
190 POKE I,A
200 NEXT I
210 REM LE MENSAGEM E CARREGA NA MEMORI
A
220 LINE INPUT "MENSAGEM:";M$
230 FOR I=1 TO LEN(M$)
240 B$=MID$(M$,I,1)
250 POKE &HE1FF+I,ASC(B$)
260 NEXT I
270 POKE &HE1FF+I,0 : REM TERMINA MENSA
GEM COM BYTE ZERO
280 REM CHAMA ROTINA EM ASSEMBLER
290 DEFUSR=&HE000 : A=USR(0)
300 PRINT:PRINT"MENSAGEM GRAVADA !!!"
310 END
```

As rotinas de cassete se dividem em dois tipos: gravação e leitura. Observe a tabela 1. Nela são relacionadas estas rotinas e dados os endereços de entrada correspondentes. Não vou descrever estas rotinas, pois são, de certa forma, complexas, envolvendo conhecimentos de linguagem de máquina e do hardware do sistema. Não pretendo, neste artigo, exigir de vocês, leitores, grande conhecimento de assembler, mas uma pequena noção é muito útil. Observe que ainda existe mais uma rotina que trata somente do estado de operação do gravador cassete.

Entre com os seguintes POKEs, pelo BASIC, para testar esta rotina:

```
POKE &HE000,&H3E
POKE &HE001,1
POKE &HE002,&HCD
POKE &HE003,&HF3
POKE &HE004.0
POKE &HE005,&HC9
```

Agora digite:

```
DEF USR = &HE000
A = USR (0)
```

Reparou no "click" do gravador sendo acionado? Experimente dar POKE 'HE001,0 e, a seguir, novamente A=USR(0). O gravador deve desligar agora. Estes POKEs, acima, somente carregam, a partir do endereço &hE000, um pequeno programa em linguagem de máquina que coloca no registrador A do Z80 um valor e chamam a rotina de operação do motor do gravador. Isto equivale às instruções BASIC: MOTOR ON e MOTOR OFF.

Não é muito difícil controlar o sistema de gravação em assembler, mas deve ser tomado muito cuidado para não se gravar ou ler informações não coerentes. Uma simples questão de tempo, ou seja, demora na leitura de um byte na fita, pode tirar todo o sentido do que o computador está lendo. As rotinas do BASIC estão escritas de forma a não permitir que coisas do tipo aconteçam, mas você também pode fazer o mesmo sem problemas.

Como exemplo de utilização destas rotinas, dois pequenos programas são fornecidos nas listagens 1 e 2. O primeiro deles gravará uma mensagem na fita cassete e o segundo lerá esta mensagem de volta. Os programas são em BASIC, contendo rotinas em linguagem de máquina em linhas DATA. As rotinas em linguagem de máquina estão descritas em assembler, respectivamente, nas listagens 3 e 4, para aqueles que querem se aprofundar mais no assunto. Agora, digite o programa da listagem 1, prepare o gravador para salvar a mensagem e rode o programa. Este programa pedirá a você para entrar uma mensagem, carregará esta mensagem na memória e, ao chamar a rotina em linguagem de máquina, passará a mensagem para fita cassete.

Digite NEW e entre com o programa da listagem 2. Este programa fará o inverso do anterior, lendo a mensagem da fita através de uma rotina em assembler e a carregará na memória. O restante do programa BASIC se encarregará de imprimir a mensagem no vídeo. Quando estiver digitado, volte um pouco a fita cassete para o início do bloco que você salvou anteriormente e prepare-o para carregar o bloco. Rode o programa e aguarde a mensagem no vídeo. Satisfeito com o resultado? Esta foi a mensagem que você digitou no primeiro programa e, apesar de ter dado um NEW, aí está ela de volta, via cassete.

Como conclusão, vemos que todos os comandos de gravação do BASIC utilizam estas rotinas de gravação do BIOS, somente lembrando que estas são bem simples, fazendo, somente, as operações mais simples, enquanto que as rotinas do interpretador BASIC já contêm testes para você poder dar um BREAK na gravação, a seleção dos dados que vão ser salvos, a gravação daquele pequeno bloco, o "header" de identificação dos programas e outras tarefas mais complexas.

Nada disto lhe impede de usá-las, como foi apresentado neste artigo, pois todas as coisas simples podem, ainda, ser bem trabalhadas e muita coisa pode ser criada em cima destas rotinas. De onde vocês acham que vieram os copiadores de programas que muitos usam? E aqueles joguinhos que só se carregam com um carregador especial contido neles mesmos?

Para aqueles que vão se aventurar, leiam as listagens 3 e 4, tentando entender o que faz cada uma das rotinas e boa sorte nas suas experiências. Em breve, voltaremos com novas explorações pelo mundo do BIOS, descobrindo o que está por trás de outras instruções do BASIC.

```
0 REM LISTAGEM 2
110 REM   O SISTEMA DE GRAVACAO CASSETE
 NO MSX
120 REM   ANDRE LUIZ FRANCO DE FREITAS
130 DATA CD,E1,00,21,00,E2,E5,CD
140 DATA E4,00,E1,77,23,FE,00,20
150 DATA F5,CD,E7,00,C9
160 REM CARREGA O PROGRAMA ASSEMBLER NA
 MEMORIA
170 FOR I=&HE100 TO &HE114
180 READ A$: A=VAL("&H"+A$)
190 POKE I,A
200 NEXT I
210 CLS : KEY OFF
220 REM CHAMA ROTINA EM ASSEMBLER
230 DEFUSR=&HE100 : A=USR(0)
240 PRINT "MENSAGEM LIDA:":PRINT:PRINT
250 E=&HE200
260 A=PEEK(E)
270 IF A=0 THEN GOTO 310*
280 PRINT CHR$(A);
290 E=E+1
300 GOTO 260
310 END
```

Menus e tabelas na screen 2

ANDRÉ L. F. DE FREITAS

A maioria dos programas existentes, principalmente os aplicativos, utilizam algum tipo de menú para que o usuário possa efetuar uma escolha entre as opções disponíveis, tornando, assim, o programa mais flexível.

Existem várias formas de se apresentar um menú, tabela ou gráfico, sendo o mais bem elaborado e de melhor resultado visual aquele no qual o menú ou tabela, ou o que desejamos apresentar, se encontra no interior de uma janela, a qual possui cor diferente do restante da tela.

Inúmeras são as possibilidades para as janelas e as instruções utilizadas para seu processamento, mesmo em Basic, são poucas e de fácil compreensão.

A principal instrução que utilizaremos é LINE com sua opção BF. Sua função é traçar uma linha, no modo gráfico, com base nas coordenadas fornecidas pelo usuário. Se a opção BF for fornecida, será traçado um retângulo, sendo que as coordenadas fornecidas correspondem aos extremos da diagonal do retângulo a ser desenhado. O retângulo também será preenchido com a cor especificada.

LINE só pode ser utilizado na tela de alta resolução, portanto, outra instrução, a SCREEN 2, será utilizada para colocar o micro no modo de alta resolução.

Digite o programa abaixo, rodando-o a seguir.

```
10 SCREEN 2:CLS
20 LINE (20,20)–(100,100), 15,BF
S30 GOTO 30
```

Na tela foi desenhada uma janela que possui cor diferente da cor de fundo da tela.

A linha 30 se faz necessária para evitar o retorno do micro ao modo de texto.

Temos agora, ao que parece, um problema: se o micro está em modo gráfico, como colocaremos texto na tela?

Todas as telas do MSX podem ser trabalhadas como arquivos para escrita. Portanto, podemos utilizar a instrução OPEN do Basic para ter acesso à escrita em uma página gráfica. Esta instrução associa um número, entre 0 e 15, a um arquivo, no nosso caso a página gráfica (GRP:), sendo este número associado à instrução PRINT, a qual se encarregará de escrever na tela gráfica.

Para posicionarmos o texto, usamos a instrução PSET, já que LOCATE não servirá por atuar somente nas telas de texto. PSET irá imprimir um ponto na tela e, após esta instrução, a posição de impressão corrente no vídeo é a logo a seguir às coordenadas de PSET. Se plotarmos um ponto com a cor de fundo da janela, este não será visível e ficaremos com as coordenadas de impressão de texto posicionadas no local que desejamos.

Após esta instrução, podemos usar o PRINT # para escrever no nosso "arquivo", ou seja, o vídeo gráfico.

Inlclua as linhas abaixo no programa anterior e rode-o.

```
5 COLOR 1,4,4
15 OPEN "GRP:" FOR OUTPUT AS
# 1
25 PSET (30,30),15
26 PRINT # 1, "TEXTO"
```

Observe a presença de texto dentro da janela. Note que com poucas linhas de programa podemos criar uma janela e escrever texto nela, mesmo em alta resolução.

O programa fornecido na listagem 1 é um exemplo de menú, onde são apresentadas várias opções, cabendo ao usuário a escolha de uma delas.

Fazendo uma leitura detalhada do programa, vamos observar as coordenadas das instruções LINE e PSET. Na listagem, estas coordenadas estão sob a forma de um número multiplicado por 8. Foram apresentadas deste modo para facilitar a idéia da tela com gráficos e texto simultâneamente. A tela gráfica possui resolução de 256x2192 pontos, enquanto a de texto, no modo SCREEN 1, possui resolução de 32x24 caracteres. Note que 32x8=256 e 24x8=192. Pense na tela gráfica como uma tela de texto de 32x24 caracteres. E, sempre que quiser converter as coordenadas dos caracteres, multiplique-as por oito, para utilizá-las nas instruções gráficas.

Observe também a subrotina que cria as janelas. A rotina foi melhorada com o acréscimo de instruções para criar uma "moldura" e um pequeno efeito de sombra nas janelas.

Leia atentamente o programa e observe o que pode ser feito com janelas, aproveitando as idéias aqui apresentadas em seus próprios programas.

Nos próximos números iremos publicar um programa em módulos que oferecerá vários recursos como calculadora, agenda telefônica, traçador de gráficos, etc.

```
170 SCREEN 2: COLOR 15,4,4: CLS
180 X1=8:Y1=7:X2=24:Y2=16:CM=1:CJ=10:GO
SUB 770
190 X=14:Y=8:CC=1:M$="MENU":GOSUB 900
200 X=11:Y=10:M$="1 - TABELA":GOSUB 900
210 X=11:Y=12:M$="2 - GRAFICO":GOSUB900
220 X=11:Y=14:M$="3 - JANELAS":GOSUB900
230 X=4:Y=1:CC=15:CJ=4:M$="Exemplos de
utilizacao de": GOSUB 900
240 X=4:Y=2:CC=15:M$="janelas em progra
mas.": GOSUB 900
250 X=4:Y=20:CC=15:M$="Por  Andre Freit
as - 1988": GOSUB 900: CC=1
260 A$=INKEY$:IF A$="" THEN 260
270 A=VAL(A$):IF A(1 OR A)3 THEN BEEP:G
OTO 260
280 ON A GOSUB 300,460,650
290 GOTO 170
310 REM   OPCAO 1 - TABELA
330 CLS
340 X1=2:Y1=6:X2=30:Y2=18:CJ=3:CF=1:GOS
UB 770
350 X=13:Y=7:M$="TABELA .GOSUB 900
360 X=4:Y=9:M$="-----------------------
-": GOSUB 900
370 X=4:Y=10:M$="! Mes ! Vendas ! Valor
 !": GOSUB 900
380 X=4:Y=11:M$="----------------------
--": GOSUB 900
390 X=4:Y=12:M$="  Jan      100    500,0
0": GOSUB 900
400 X=4:Y=13:M$="  Fev       80    400,0
0": GOSUB 900
410 X=4:Y=14:M$="  Mar      150    750,0
0": GOSUB 900
420 X=4:Y=15:M$="  Abr      190    950,0
0": GOSUB 900
430 X=4:Y=16:M$="----------------------
--": GOSUB 900
440 FOR I=1 TO 4000: NEXT I
450 RETURN
470 REM   OPCAO 2 - GRAFICO
490 CLS
500 X1=2:Y1=2:X2=30:Y2=22:CJ=14:CM=1: G
OSUB 770
510 X=12:Y=4:M$="GRAFICO": GOSUB 900
520 X=4:Y=8:M$="Vendas": GOSUB 900
530 X=4:Y=10:M$="200 -": GOSUB 900
540 X=4:Y=12:M$="150 -": GOSUB 900
550 X=4:Y=14:M$="100 -": GOSUB 900
560 X=4:Y=16:M$=" 50 -": GOSUB 900
570 X=4:Y=18:M$="  0 -----------------
--":GOSUB 900
580 X=4:Y=20:M$="      Jan  Fev  Mar  Ab
r": GOSUB 900
590 X1=9:Y1=14:X2=12:Y2=18:CJ=7:GOSUB 7
70
600 X1=14:Y1=16:X2=17:Y2=18:CJ=8:GOSUB
770
610 X1=19:Y1=12:X2=22:Y2=18:CJ=2:GOSUB
770
620 X1=24:Y1=10:X2=27:Y2=18:CJ=10:GOSUB
 770
630 FOR I=1 TO 4000: NEXT I
640 RETURN
660 REM   OPCAO 3 - JANELAS
680 CLS
690 X=13:Y=2:CC=15:M$="JANELAS": GOSUB
900
700 X1=4:Y1=4:X2=8:Y2=22:CJ=6:GOSUB 770
710 X1=6:Y1=6:X2=18:Y2=10:CJ=2:GOSUB8770
720 X1=20:Y1=4:X2=30:Y2=16:CJ=13:GOSUB
770
730 X1=11:Y1=18:X2=28:Y2=22:CJ=7:GOSUB
770
740 X1=13:Y1=8:X2=17:Y2=20:CJ=14:GOSUB
770
750 FOR I=1 TO 4000: NEXT I
760 RETURN
770 REM  SUBROTINA QUE CRIA JANELAS
790 REM  PARAMETROS:
800 REM      X1,Y1 - COORD. INICIAIS
810 REM      X2,Y2 - COORD. FINAIS
820 REM      CJ    - COR DA JANELA
830 REM      CM    - COR DA MOLDURA
850 LINE (X1*8,Y1*8)-(X2*8,Y2*8),CJ,BF
860 LINE (X1*8,Y1*8)-(X2*8,Y2*8),CM,B
870 LINE (X1*8+3,Y1*8-3)-(X2*8+3,Y1*8),
CM,BF
880 LINE (X2*8,Y1*8)-(X2*8+3,Y2*8-3),CM
,BF
890 RETURN
920 REM  SUBROTINA QUE ESCREVE
930 REM  NA PAGINA DE ALTA RESOLUCAO
950 REM  PARAMETROS:
960 REM      X,Y   - COORD. MENSAGEM
970 REM      CJ    - COR DE FUNDO
980 REM      CC    - COR CARACTERES
990 REM      M$    - MENSAGEM
1010 OPEN "GRP:" FOR OUTPUT AS #1
1020 PSET (X*8,Y*8),CJ
1030 COLOR CC
1040 PRINT #1, M$
1050 CLOSE #1
1060 RETURN
```

JOGOS & HIGH SCORES

Nesta seção, publicaremos os high scores de jogos e dicas de como melhorar o seu desempenho e ajudá-lo a salvar todas as princesas, planetas e demais seres que vivem em perigo, implorando nossa ajuda.

HIGH SCORES

ALIEN 8	49%
BOULDER DASH	55.848
BUCK ROGERS	310.900
CHORO Q	42.380
CIRCUS CHARLIE	1.198.400
ELIDON	94%
FLIGHT DECK	6.410
GALAGA	850.000
INTERNATIONAL KARATE	999.999
KING'S VALLEY	5.642.600
KNIGHT MARE	238.020
LAZY JONES	149.650
OH SHIT	76.250
POLAR STAR	289.900
PYRAMIDE WARP	820.758
RIVER RAID	73.450
ROAD FIGHTER	998.675
ROLERBALL	3.120.180
SCION	501.100
SOCCER	40-0
SUPER COBRA	6.348.460
SWEET ACORN	9.990
TIME BANDITS	176.050

Se você já obteve um high score mais alto dos aqui apresentados, ou em qualquer outro jogo, envie-nos sua pontuação acompanhada de alguma comprovação, como fotografia da tela ou descrição das fases percorridas, para que possamos publicá-la, juntamente com o seu nome.

Se você é fera, nada mais justo do que o seu nome constar na seção de High Scores de CPU.

Os jogos que oferecem facilidades adicionais, com tiro múltiplo, vidas infinitas, etc., só serão considerados na sua versão original.

```
0 REM
1 REM REVISTA CPU - MAIO 1988
2 REM AUF MONTY
3 REM LEITOR PARA MIL VIDAS
10 CLS:KEYOFF:COLOR10,1,1:SCREEN2
20 OPEN"GRP:"FOROUTPUTAS#1
30 FORX=1TO2:PSET(29+1+X,106),POINTSTEP
(0,0):PRINT#1,+CHR$(34)"AUF WIEDERSEHEN
 MONTY"+CHR$(34):NEXTX
40 COLOR11:FORV=1TO2:PSET(50+1+V,140),1
:PRINT#1,"1 -> IMUNIDADE TOTAL.":NEXTV
41 COLOR11:FORE=1TO2:PSET(50+1+E,160),P
OINTSTEP(0,0):PRINT#1,"2 -> VIDAS INFIN
ITAS.":NEXTE
42 COLOR11:FORU=1TO2:PSET(50+1+U,180),P
OINTSTEP(0,0):PRINT#1,"3 -> NORMAL.":NE
XTU:CLOSE#1
50 A$=INPUT$(1)
60 IF A$="1"THEN100
70 IF A$="2"THEN130
80 IF A$="3"THEN150
90 GOTO 50
100 BLOAD"MONTY1",R:BLOAD"MONTY2",R:BLO
AD"MONTY3",R
110 BLOAD"MONTY4",R:BLOAD"MONTY5",R
120 BLOAD"MONTY6":DEFUSR=&H8700:POKE&HA
C60,240:POKE&H9808,0:POKE&HA970,240:A=U
SR(0)
130 BLOAD"MONTY1",R:BLOAD"MONTY2",R:BLO
AD"MONTY3",R:BLOAD"MONTY4",R
140 BLOAD"MONTY5",R:BLOAD"MONTY6":DEFUS
R=&H8700:POKE&HA970,240:A=USR(1)
150 BLOAD"MONTY1",R:BLOAD"MONTY2",R:BLO
AD"MONTY3",R:BLOAD"MONTY4",R
160 BLOAD"MONTY5",R:BLOAD"MONTY6",R
```

```
0 REM
1 REM ZANAC
2 REM LEITOR PARA VIDAS INFINITAS
10 CLS:KEYOFF:COLOR10,1,1:SCREEN2
20 OPEN"GRP:"FOROUTPUTAS#1
30 FORX=1TO2:PSET(50+1+X,106),POINTSTEP
(200,5):PRINT#1,+CHR$(34)"ZANAC"+CHR$(3
4):NEXTX
41 COLOR11:FORV=1TO2:PSET(50+1+V,160),P
OINTSTEP(0,0):PRINT#1,"1 -> VIDAS INFIN
ITAS.":NEXTV
42 COLOR11:FORU=1TO2:PSET(50+1+U,180),P
OINTSTEP(0,0):PRINT#1,"2 -> NORMAL.":NE
XTU:CLOSE#1
50 A$=INPUT$(1)
60 IF A$="1"THEN100
70 IF A$="2"THEN120
90 GOTO 50
100 BLOAD"CAS:":POKE&H9654,0
110 DEFUSR=&HD000:A=USR(0)
120 BLOAD"CAS:",R
130 BLOAD"CAS:",R
```

```
0 REM
1 REM GALAGA
2 REM LEITOR PARA MIL VIDAS
10 CLS:KEYOFF:COLOR10,1,1:SCREEN2
20 OPEN"GRP:"FOROUTPUTAS#1
30 FORX=1TO2:PSET(50+1+X,106),POINTSTEP
(200,5):PRINT#1,+CHR$(34)"GALAGA"+CHR$(
34):NEXTX
41 COLOR11:FORV=1TO2:PSET(50+1+V,160),P
OINTSTEP(0,0):PRINT#1,"1 -> VIDAS INFIN
ITAS.":NEXTV
42 COLOR11:FORU=1TO2:PSET(50+1+U,180),P
OINTSTEP(0,0):PRINT#1,"2 -> NORMAL.":NE
XTU:CLOSE#1
50 A$=INPUT$(1)
60 IF A$="1"THEN100
70 IF A$="2"THEN150
90 GOTO 50
100 BLOAD"CAS:",R
110 BLOAD"CAS:"
120 POKE &H9152,0
130 DEFUSR=PEEK(&HFCC0)*256+PEEK(&HFCBF
)
140 A=USR(0)
150 BLOAD"CAS:",R:BLOAD"CAS:",R
```

```
0 REM
1 REM PROFANATION
2 REM LEITOR PARA MIL VIDAS
10 CLS:KEYOFF:COLOR10,1,1:SCREEN2
20 OPEN"GRP:"FOROUTPUTAS#1
30 FORX=1TO2:PSET(50+1+X,106),POINTSTEP
(200,5):PRINT#1,+CHR$(34)"PROFANATION"+
CHR$(34):NEXTX
41 COLOR11:FORV=1TO2:PSET(50+1+V,160),P
OINTSTEP(0,0):PRINT#1,"1 -> VIDAS INFIN
ITAS.":NEXTV
42 COLOR11:FORU=1TO2:PSET(50+1+U,180),P
OINTSTEP(0,0):PRINT#1,"2 -> NORMAL.":NE
XTU:CLOSE#1
50 A$=INPUT$(1)
60 IF A$="1"THEN100
70 IF A$="2"THEN150
90 GOTO 50
100 BLOAD"CAS:",R:BLOAD"CAS:",R:BLOAD"C
AS:":POKE&HC084,240
110 BLOAD"CAS:",R:BLOAD"CAS:",R:BLOAD"C
AS:",R:BLOAD"CAS:",R:BLOAD"CAS:",R
```

```
0 REM
1 REM BOUNDER
2 REM LEITOR PARA MIL VIDAS
10 CLS:KEYOFF:COLOR10,1,1:SCREEN2
20 OPEN"GRP:"FOROUTPUTAS#1
30 FORX=1TO2:PSET(50+1+X,106),POINTSTEP
(200,5):PRINT#1,+CHR$(34)"BOUNDER"+CHR$
(34):NEXTX
41 COLOR11:FORV=1TO2:PSET(50+1+V,160),P
OINTSTEP(0,0):PRINT#1,"1 -> VIDAS INFIN
ITAS.":NEXTV
42 COLOR11:FORU=1TO2:PSET(50+1+U,180),P
OINTSTEP(0,0):PRINT#1,"2 -> NORMAL.":NE
XTU:CLOSE#1
50 A$=INPUT$(1)
60 IF A$="1"THEN100
70 IF A$="2"THEN110
90 GOTO 50
100 BLOAD"CAS:",R:BLOAD"CAS:":POKE&H8C8
7,200:DEFUSR=&H8700:A=USR()
110 BLOAD"CAS:",R:BLOAD"CAS:",R:BLOAD"C
AS:",R:BLOAD"CAS:",R:BLOAD"CAS:",R
```

CURSO DE MÚSICA – Volume 1
Editora Aleph - 144 Páginas - 14x21 cm
Barbieri, Piazzi

Não tem sentido tentar se aprender teoria musical sem se tocar um instrumento. A maioria das pessoas, porém, por não disporem de tempo e paciência suficientes para adquirir uma razoável habilidade num instrumento qualquer, marginalizam-se do maravilhoso mundo da música. Neste caso limitam-se, se muito, ao papel de passivos ouvintes.

Com o advento do MSX, porém, este ouvintes passivos passam a dispor de um eficientíssimo instrumento musical de 3 vozes que vai lhes permitir aprender música interagindo (com um instrumento musical tradicional) sem que haja necessidade de um adestramento psicomotor.

O objetivo principal deste livro não é o ensino de programação, apesar disso acabar ocorrendo de forma suave e didática, mas sim o aprendizado, por parte do leitor, da teoria musical em si, transformando-o num ouvinte ativo e em certos casos, até num compositor de talento!

100 DICAS PARA MSX
TÉCNICAS E TRUQUES DE PROGRAMAÇÃO
Editora Aleph - 191 Páginas - 14x21 cm
Renato da Silva Oliveira

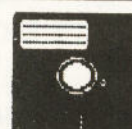
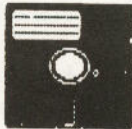
Visando facilitar a programação em Basic, poupando tempo e esforço, o livro apresenta listagens de programas que poderão ser utilizadas pelo programador em sub-rotinas.

Dividido em 7 capítulos (teclado, vídeo, som, cassete, impressora, drive e processamento), torna-se indispensável para quem deseja aproveitar todos os recursos disponíveis do MSX, explorando assuntos que não se encontram nos manuais que acompanham o equipamento.

DRIVES LEOPARD DE 3 1/2"
NOVOS HORIZONTES PARA O SEU MSX
Editora Aleph – 120 Páginas
13x18 cm
Carvalho Jr., Oliveira, Pjazzi

Destinado aos usuários de drives de 3 1/2", ou de qualquer drive de 5 1/4", ou possuidores de acionador Leopard modelo DT 300 ou DT 350, da Technoahead, o livro fornece, de maneira objetiva e clara, informações necessárias para operação com o sistema operacional MSXDOS e Disk Basic.

Todas as instruções do MSXDOS e do Disk Basic são comentadas, sendo fornecidos exemplos e comentados os possíveis casos de erro que poderão ocorrer.

Uma grande ênfase é dada à programação com os comandos específicos do Disk Basic, e do MSXDOS, dando total suporte ao programador.

Jawbrake

CÉSAR MATTOS

Este jogo foi baseado no velho Pac Man.

Para fazer pontos, você deverá comer todas as vitaminas espalhadas pela tela, tomando cuidado com os guardiães que tentam impedir, a todo o custo, o seu objetivo.

Preste atenção na parte central da tela e tente apanhar as super vitaminas que de vez em quando aparecem.

Para jogar, use as setas.

```
20 TIME=0:KEYOFF:CLS:CLEAR1000:SCREEN1,
2:COLOR15,1,1:CLS:ONSPRITEGOSUB920:OPEN
"GRP:"FOROUTPUTAS#1:STRIG(0)ON
30 A$="":B$="":RESTORE1100:FORX=1TO16:R
EADD$:A$=A$+CHR$(VAL("&B"+LEFT$(D$,8)))
40 B$=B$+CHR$(VAL("&B"+RIGHT$(D$,8)))
50 NEXTX:SPRITE$(0)=A$+B$
60 A$="":B$="":RESTORE1270:FORX=1TO16:R
EADD$:A$=A$+CHR$(VAL("&B"+D$)):NEXTX:SP
RITE$(1)=A$
70 'VARIAVEIS
80 TT=0:TP=0:CH=3
90 M=154:N=36:'COORDENADAS
100 'APRESENTACAO
110 GOSUB1480
120 SCREEN2
130 PRESET(84,76):COLOR12:PRINT#1,"ARCA
DIA SOFT":PRESET(96,90):COLOR7:PRINT#1,
"apresenta"
135 PRESET(85,77):COLOR12:PRINT#1,"ARCA
DIA SOFT":PRESET(97,91):COLOR7:PRINT#1,
"apresenta"
138 LINE(78,70)-(185,70):LINE(78,105)-(
185,105):LINE(78,70)-(78,105):LINE(185,
70)-(185,105)
139 LINE(78,71)-(185,71):LINE(78,106)-(
185,106):LINE(79,71)-(79,106):LINE(186,
71)-(186,106)
140 FORV=1TO2:FORX1=60TO180:PUTSPRITE4,
(X1,50):NEXTX1
150 FORX=70TO106:PUTSPRITE4,(180,X),12,
0:NEXTX
160 FORX1=150TO80STEP-1:PUTSPRITE4,(X1,
160),12,0:NEXTX1
170 FORX=106TO60STEP-1:PUTSPRITE4,(65,X
),12,0:NEXTX
180 NEXTV
190 PUTSPRITE4,(-30,255),12,0:IFPLAY(0)
THENGOTO190
200 SCREEN3:COLOR,1:CLS
210 LINE(15,0)-(235,230),2,B
220 FORGG=1TO2:Y=0:Y1=0:A=RND(1)*13+2:I
FA=7THENA=6
230 FORX=1TO10
240 COLORA:Y1=Y1+6:Y=Y+14:PSET(6+Y+Y1,Y
1+70),1:PRINT#1,MID$("Jawbreake",X,1)
250 SOUND7,56:SOUND8,15:FORK=1TO20STEP4
:SOUND0,K*5:NEXTK
260 PUTSPRITE4,(6+Y+Y1,100),A,0
270 NEXTX:A=RND(1)*13+2:IFA=7THENA=6
280 Y=Y1+Y+19:FORX=10TO1STEP-1
290 Y=Y-20:PSET(6+Y,70),1:COLORA:PRINT#
1,MID$("Jawbreake",X,1)
300 PUTSPRITE4,(6+Y,100),A,0
310 FORK=20TO1STEP-4:SOUND0,K*5:NEXTK:N
EXTX
320 ONSTRIGGOSUB1470
330 NEXTGG
340 GOSUB1480:WIDTH30:SCREEN1:COLOR1,7,
2:LOCATE4,5:PRINTSTRING$(22,223):LOCATE
6,6:PRINT"J A W B R E A K E R":LOCATE4,
7:PRINTSTRING$(22,220):PUTSPRITE4,(122,
80),13,0
350 PUTSPRITE4,(122,80),6,0
360 LOCATE1,22:PRINT"Pres. barra de esp
ao p/jogar";:FORX=1TO2000:NEXTX:COLOR,
,1:GOTO200
370 'CENARIO
380 COLOR15,1:SCREEN2:SPRITEON:STRIG(0)
OFF
390 COLOR7:PRESET(40,1):PRINT#1,"JAWBRE
AKER":COLOR12:PRESET(150,1):PRINT#1,STR
ING$(CH,CHR$(249)):COLOR14:PRESET(60,17
5):PRINT#1,"ARCADIA SOFT 1986":COLOR15
395 COLOR7:PRESET(41,1):PRINT#1,"JAWBRE
AKER":COLOR12:PRESET(150,1):PRINT#1,STR
ING$(CH,CHR$(249)):COLOR14:PRESET(60,17
6):PRINT#1,"ARCADIA SOFT 1986":COLOR15
400 FORX=1TO3:LINE(30-X,12-X)-(220-X,17
2-X),12,B:NEXTX
410 LINE(50,30)-(200,30),4
420 LINE(50,50)-(200,50),4
430 LINE(50,70)-(200,70),4:LINE(50,90)-
(200,90),4:LINE(50,110)-(200,110),4:LIN
```

```
E(50,130)-(200,130),4:LINE(50,150)-(200
,150),4
440 FORX=18TO160STEP20:PRESET(50,X):PRI
NT#1,STRING$(19,CHR$(196)):NEXTX
445 ONINTERVAL=1500GOSUB1450
450 'COMANDO DO JOGO
460 INTERVALON
470 V=RND(1)*12:W=RND(1)*12
480 FORX=30TO203STEP8:X1=X:X2=211-X+20
490 IFV<6THENGOSUB840
500 IFW<4THENGOSUB850
510 IFW>8THENGOSUB860
520 IFV>3THENGOSUB870
530 IFW<10THENGOSUB880
540 IFV<8THENGOSUB890
550 IFV>5THENGOSUB900
560 IFV<6THENGOSUB910
570 GOSUB8720:NEXTX
580 FORX=30TO203STEP8:X2=X:X1=211-X+20
590 IFV<6THENGOSUB8840
600 IFW<4THENGOSUB8850
610 IFW>8 THEN GOSUB 840
620 IFV>3THENGOSUB8870
630 IFW<10THENGOSUB8880
640 IFV<8THENGOSUB8890
650 IFV<5THENGOSUB8900
660 IFV<6THENGOSUB8910
670 GOSUB720
680 NEXTX
690 AA=RND(1)*4
700 FORX=2TO10:PUTSPRITEX,(-20,255),9,1
:NEXTX:IFAA<3THENGOTO450
710 GOTO580
720 'MOV. JAW
730 INTERVALON:A=STICK(0):IFA=7ANDN>36T
HENN=N-8
740 IFN=124ANDM=74ANDFLAG=1THENGOTO780
750 IFA=3ANDN<203THENN=N+8
760 IFA=1ANDPOINT(N+7,M-4)<>4ANDPOINT(N
,M-4)<>4ANDM>17THENM=M-20
770 IFA=5ANDPOINT(N+7,M+16)<>4ANDPOINT(
N,M+16)<>4ANDM<150THENM=M+20
780 PUTSPRITE1,(N,M),9,1
790 IFN=124ANDM=74ANDFLAG=1THENPLAY"V13
L32ABCDDCBA":TT=TT+100:PT=PT+100:GOSUB1
460
800 IFPOINT(N+3,M+8)=15ORPOINT(N+3,M+8)
=7THENTP=TP+1:PT=PT+10:PRESET(N,M+4):CO
LOR1:PRINT#1,CHR$(219):COLOR15:PLAY"V13
SOM1000O4C#":TT=TT+10:IFTP=151THENGOSUB
1480:GOSUB1050
810 IFTT>=5000THENTT=0:CH=CH+1:PLAY"V15
T255O4L8DL4BL8GL8DL4CL8AL1GL32":COLOR7:
PRESET(40,1):PRINT#1,"JAWBREAKER":COLOR
12:PRESET(150,1):PRINT#1,STRING$(CH,CHR
$(249)):COLOR15
820 RETURN
830 'IMPRESS.DOS INIMIGOS
840 PUTSPRITE3,(X2,30),10,0:RETURN
850 PUTSPRITE4,(X1,10),12,0:RETURN
860 PUTSPRITE5,(X1,50),13,0:RETURN
870 PUTSPRITE6,(X2,70),8,0:RETURN
880 PUTSPRITE7,(X1,90),2,0:RETURN
890 PUTSPRITE8,(X2,110),7,0:RETURN
900 PUTSPRITE9,(X1,130),9,0:RETURN
910 PUTSPRITE10,(X2,150),14,0:RETURN
920 IFPLAY(0)THEN920
930 SOUND7,56:SPRITEOFF:SOUND8,15:SOUND
1,0:FORY=50TO200STEP5:SOUND0,Y+RND(1)*2
5:PUTSPRITE1,(N,M),RND(1)*7,1:NEXTY:SOU
ND8,0
940 CH=CH-1:IFCH=0THENGOTO990
950 COLOR1:PRESET(40,1):PRINT#1,STRING$
(22,CHR$(219))
960 N=124:M=74:GOSUB8720
970 COLOR7:PRESET(40,1):PRINT#1,"JAWBRE
AKER":COLOR12:PRESET(150,1):PRINT#1,STR
ING$(CH,CHR$(249)):COLOR15
980 FORS=3TO10:PUTSPRITES,(-20,255),1,0
:NEXTS:SPRITEON:RETURN
990 'ROT.FINALIZACAO
1000 PRESET(82,78):COLOR1:PRINT#1,STRIN
G$(12,CHR$(219)):COLOR1:PRESET(80,96):P
RINT#1,STRING$(12,CHR$(219)):PRESET(84,
78):COLOR10:PRINT#1,"FIM DE JOGO":PRESE
T(80,100):PRINT#1,USING"####";PT;:PRINT
#1,"PONTOS"
1010 PLAY"V15L1605CC04BAGFEDCCCCC","V13
L1604CCDEFGAB05CCCCC","V1403L4CCCL16C"
1020 FORY=1TO10:PUTSPRITEY,(0,0),1,0:NE
XTY:FORY=1TO1000:NEXTY:CLOSE
1030 IFPLAY(0)THEN1030
1040 RUN
1050 'GO. TP=152
1060 INTERVALOFF:SPRITEOFF:FORY=2TO10:P
UTSPRITEY,(-20,255),9,1:NEXTY
1070 PRESET(74,77):COLOR12:PRINT#1,PT;"
PONTOS":FORX=1TO200:NEXTX:COLOR1:PRESET
(74,77):PRINT#1,STRING$(15,CHR$(219)):C
OLOR15
1080 TP=0:N=124:M=74:PUTSPRITE1,(N,M),9
,1
1090 FORY=18TO160STEP20:PRESET(50,Y):PR
INT#1,STRING$(19,CHR$(196)):NEXTY:SPRIT
EON:INTERVALON:RETURN
1100 DATA 0000000000000000
1110 DATA 0001111111111000
1120 DATA 0011111111111100
1130 DATA 0111111111111110
1140 DATA 1110001111000111
1150 DATA 1110101111010111
1160 DATA 1110001111000111
1170 DATA 1110001111000111
1180 DATA 1111111111111111
1190 DATA 1111111111111111
```

```
1200 DATA 1111111111111111
1210 DATA 1111100000011111
1220 DATA 1111100000011111
1230 DATA 1111111111111111
1240 DATA 0101010101010101
1250 DATA 1010101010101010
1260
1270 DATA 11111111
1280 DATA 11111111
1290 DATA 10011001
1300 DATA 10111011
1310 DATA 10011001
1320 DATA 11111111
1330 DATA 1000001
1340 DATA 11000011
1350 DATA 11100111
1360 DATA 11111111
1370 DATA 11111111
1380 DATA 01010100
1390 DATA 01010100
1400 DATA 00000000
1410 DATA 00000000
1420 DATA 00000000
1430 BEEP:GOTO1430
1440 DATA 00000000
1450 COLOR7:PRESET(122,78):PRINT#1,CHR$
(1)CHR$(66);FLAG=1:ONINTERVAL=300GOSUB1
460:COLOR15:RETURN
1460 COLOR1:PRESET(122,78):PRINT#1,CHR$
(219):FLAG=0:ONINTERVAL=1500GOSUB1450:R
ETURN
1470 COLOR,,1:PLAY"L32C16DEFGAB05C":RET
URN370
1480 SOUND7,56:PLAY"T250","T250":PLAY"D
1M5000L4","S9M400L4"
1490 PLAY"04R1","03C8.E8.G4.G4."
1500 PLAY"04C8.E8.G4.G4.","R1"
1510 PLAY"04F4.E4.D4.","03F4.E4.D4."
1520 PLAY"04G4.G4.E4.","03E4.E4.C4"
1530 RETURN
```